**MANUAL BOOK**

**SINEMA: Landslide Inventory Mapping**

**A mountain with a pin on it

AI-generated content may be incorrect.**A screenshot of a computer

Description automatically generated

**BADAN RISET DAN INOVASI NASIONAL**

**2025**

**INFORMASI APLIKASI**

Hak cipta yang diusulkan berupa perangkat lunak aplikasi (software application) dengan nama dan judul:

“SINEMA: Landslide Inventory Mapping”

Aplikasi ini digunakan untuk mendokumentasikan dan memetakan kejadian tanah longsor secara terstruktur. Pengguna dapat menggambar area longsor menggunakan polygon di peta, mengisi data detail (lokasi, tanggal, kondisi area, kondisi longsor, dampak, faktor penyebab, mekanisme), serta mengunggah foto pendukung setiap kejadian. Hasilnya tersimpan di basis data PostGIS sehingga dapat dianalisis atau diekspor kapan saja.

Aplikasi ini merupakan pengembangan dari sistem pemetaan web berbasis Leaflet.js dan Flask, dengan tambahan:

* Manajemen pengguna (JWT authentication)
* Penyimpanan atribut detil longsor
* Dukungan unggah foto untuk setiap entri

**A screenshot of a login screen

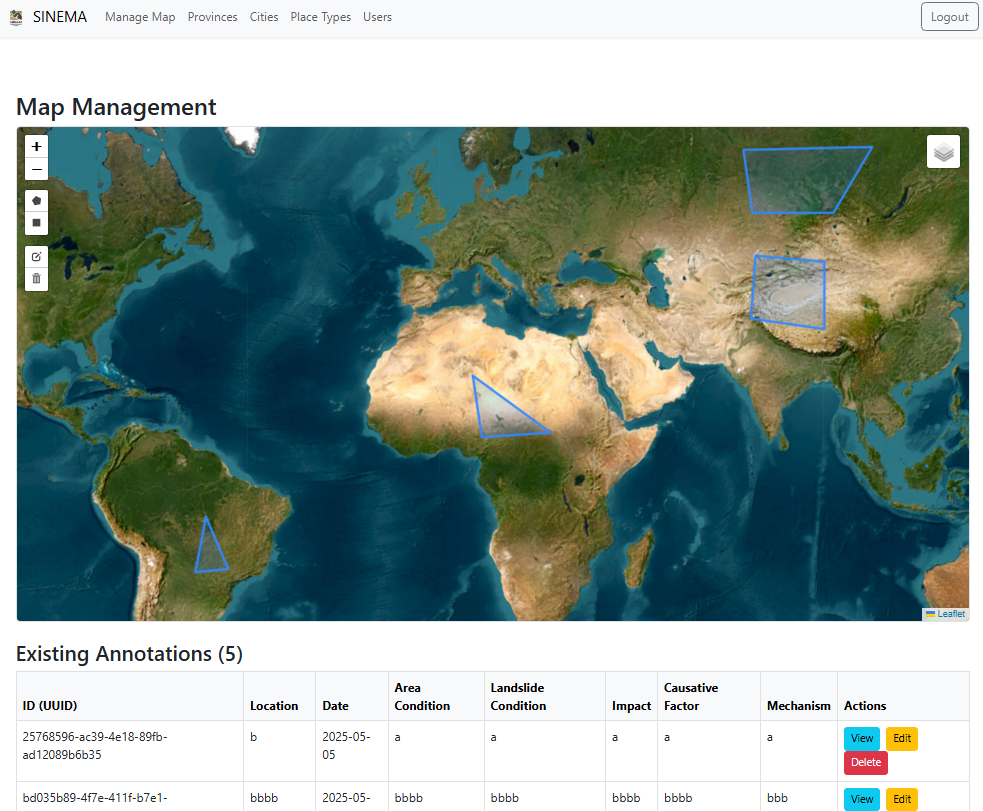
AI-generated content may be incorrect.**

**A screenshot of a login form

AI-generated content may be incorrect.**

**A screenshot of a graph

AI-generated content may be incorrect.**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**1. Keunggulan Aplikasi**

**Pemetaan Interaktif Berbasis GIS**

Dengan integrasi Leaflet, pengguna dapat langsung menandai titik lokasi pengukuran di peta dunia nyata, memudahkan visualisasi sebaran data unsur hara tanpa memerlukan perangkat lunak SIG khusus. Dibangun dengan Bootstrap 5, dashboard dan form-form aplikasi otomatis menyesuaikan tampilan pada desktop, tablet, maupun ponsel, sehingga dapat digunakan di lapangan langsung lewat tablet atau smartphone. Dashboard menyediakan filter provinsi dan jenis tanaman, serta chart interaktif (pie/bar) untuk melihat distribusi titik per wilayah atau per jenis tanaman dengan sekali klik—mempercepat analisis awal tanpa keluar dari browser.

**Pengukuran Multi-Parameter Tanah**

Selain koordinat dan metadata lahan, PUNA-NPK mendukung pencatatan mendalam (soil\_depth) serta kadar Nitrogen, Phosporus, dan Kalium dalam satu entri—menggantikan aplikasi satu-parameter yang hanya mencatat satu unsur saja.

**Otentikasi dan Keamanan Data**

Semua endpoint dilindungi JWT (JSON Web Token), sehingga hanya pengguna terdaftar dengan token valid yang dapat mengakses, menambah, atau mengubah data—menjaga kerahasiaan dan integritas hasil pengukuran.

**Dukungan Unggah Foto Lapangan**

Fitur upload foto memungkinkan dokumentasi kondisi lapangan (contoh: tekstur tanah, kondisi tanaman) tersimpan bersama titik data, membantu validasi dan audit kualitas pengambilan sampel.

**Ekstensibilitas dan Open-Source**

* **Backend:** Flask + SQLAlchemy + GeoAlchemy2 — mudah diperluas dan dipelihara.
* **Frontend:** Bootstrap, Leaflet, Chart.js — komponen terbuka yang banyak dokumentasi dan komunitas.
* Strukturnya modular, memudahkan penambahan parameter baru (misalnya pH tanah, kelembaban).

**Pengelolaan Master Data Terpusat**

CRUD untuk Provinsi, Kota, Crop Types, Soil Types, dan Fertilization Types memastikan konsistensi data master di seluruh aplikasi tanpa harus langsung intervensi basis data.

**2. Kebaruan Aplikasi**

Aplikasi SINEMA memperkenalkan sejumlah inovasi tersendiri jika dibandingkan dengan solusi pemetaan longsor atau inventarisasi geospasial yang sudah ada:

1. Form Entri Terpadu dalam Satu Modal
   * Berbeda dari banyak aplikasi GIS web lain yang memisahkan geometri dan atribut ke beberapa dialog atau panel, SINEMA menyajikan semua kolom data (lokasi, tanggal, kondisi, faktor penyebab, mekanisme) dan unggah foto dalam satu modal terpadu.
   * Hal ini meminimalkan klik dan mengurangi risiko data terlewat saat pengguna membuat anotasi polygon.
2. Unggah Foto Multi-File Terintegrasi
   * Aplikasi pembanding seperti Google Earth Engine atau QField umumnya hanya mendukung lampiran satu gambar atau memerlukan plugin tambahan. SINEMA built-in mendukung multiple file upload langsung dari browser, menyimpan banyak foto sekaligus untuk satu entri, lalu menampilkannya sebagai thumbnail di modal.
3. Master Data Dinamis (Provinsi, Kota, Tipe Lokasi)
   * Tidak seperti aplikasi desktop GIS yang memaksa pengguna memodifikasi file konfigurasi atau database lokal, SINEMA menghadirkan CRUD antarmuka web untuk Provinsi, Kota, dan Tipe Lokasi.
   * Dropdown di form annotasi selalu sinkron dengan master data terbaru, tanpa perlu redeploy kode.
4. Dashboard Analitik Real-Time dengan Minimap
   * Banyak aplikasi pemetaan hanya menyediakan peta kosong atau grafik terpisah. SINEMA menampilkan mini-map dari poligon terbaru langsung di dashboard, digabung dengan kartu ringkasan (counts) dan chart (Chart.js).
   * Ini memudahkan pemantauan cepat area rawan tanpa harus menavigasi ke halaman map.

**3. Spesifikasi Kebutuhan Aplikasi**

3.1 Bahasa Pemrograman

* Utama
  + Python 3.8+ (Flask 2.x, Flask-SQLAlchemy, GeoAlchemy2, Flask-JWT-Extended)
* Pendukung
  + HTML5 & CSS3 (Bootstrap 5.3.0)
  + JavaScript (ES6+) (Leaflet 1.9.4, Leaflet.draw 1.0.4, Chart.js 3.x)

3.2 Syarat Minimum Komputer

* Server
  + CPU: Dual-core Intel Core i3 (atau setara)
  + RAM: ≥ 4 GB
  + Storage: ≥ 50 GB SSD
  + Konektivitas: ≥ 100 Mbps
  + Sistem operasi: Ubuntu 20.04 LTS (atau Windows Server 2019)
* Client
  + Browser modern (Chrome/Firefox/Edge terbaru)
  + Resolusi layar ≥ 1024×768

3.3 Ketergantungan terhadap Software Lain

* Database Server: PostgreSQL 13+ dengan ekstensi PostGIS 3.x
* Library Python:
  + psycopg2-binary (driver PostgreSQL)
  + shapely (manipulasi geometri)
* Web Server / WSGI:
  + Gunicorn 20.x (produksi) atau Flask built-in server (development)

3.4 Jenis Server

* HTTP Server: Nginx 1.18 atau Apache 2.4 (dengan proxy\_pass ke Gunicorn)
* WSGI: Gunicorn atau uWSGI

3.5 Jenis Database

* PostgreSQL 13+
* PostGIS 3.x untuk tipe data spasial (GEOMETRY, Polygon)

3.6 Framework & Library Lain

* Bootstrap 5.3.0 (komponen UI responsif)
* Leaflet.js 1.9.4 (peta interaktif)
* Leaflet.draw 1.0.4 (tools gambar poligon)
* Chart.js 3.x (grafik analitik)
* Flask-JWT-Extended (autentikasi token-based)

**4. Algoritma Aplikasi**

A diagram of a computer

AI-generated content may be incorrect.

|  |
| --- |
| // PSEUDOCODE SINEMA: Landslide Inventory Mapping  // Entry point  FUNCTION MAIN()  token ← LOGIN()  IF token IS NULL  EXIT APPLICATION  ENDIF  LOOP  choice ← SHOW\_DASHBOARD(token)  SWITCH choice  CASE "Manage Map":  MANAGE\_MAP(token)  CASE "Manage Provinces":  MANAGE\_PROVINCES(token)  CASE "Manage Cities":  MANAGE\_CITIES(token)  CASE "Manage Place Types":  MANAGE\_PLACE\_TYPES(token)  CASE "Manage Users":  MANAGE\_USERS(token)  CASE "Logout":  BREAK LOOP  END SWITCH  END LOOP  END FUNCTION  // 1. Authentication  FUNCTION LOGIN()  WHILE TRUE  credentials ← PROMPT\_USER\_FOR\_CREDENTIALS()  response ← HTTP\_POST("/login", credentials)  IF response.status == 200  RETURN response.access\_token  ELSE  SHOW\_ERROR("Login failed, coba lagi.")  ENDIF  END WHILE  END FUNCTION  // 2. Dashboard display and menu  FUNCTION SHOW\_DASHBOARD(token)  // Fetch counts and data  annotations ← HTTP\_GET("/annotations", token).features  provinces ← HTTP\_GET("/provinces", token)  cities ← HTTP\_GET("/cities", token)  types ← HTTP\_GET("/place\_types", token)  users ← HTTP\_GET("/users", token)  // Render summary cards  DISPLAY\_CARD("Annotations", LENGTH(annotations))  DISPLAY\_CARD("Provinces", LENGTH(provinces))  DISPLAY\_CARD("Cities", LENGTH(cities))  DISPLAY\_CARD("Place Types", LENGTH(types))  DISPLAY\_CARD("Users", LENGTH(users))  // Render charts  RENDER\_BAR\_CHART("By Province", annotations, provinces)  RENDER\_PIE\_CHART("By Place Type", annotations, types)  // Render minimap of latest annotation  IF LENGTH(annotations) > 0  latest ← annotations[LENGTH(annotations)-1]  RENDER\_MINIMAP("minimap", latest.geometry)  ENDIF  // Show navigation menu  RETURN PROMPT\_USER\_MENU([  "Manage Map",  "Manage Provinces",  "Manage Cities",  "Manage Place Types",  "Manage Users",  "Logout"  ])  END FUNCTION  // 3. Manage Map annotations  FUNCTION MANAGE\_MAP(token)  WHILE TRUE  annotations ← HTTP\_GET("/annotations", token).features  DISPLAY\_MAP(annotations)  DISPLAY\_TABLE(annotations)  action ← PROMPT\_TABLE\_ACTION(["Create", "Edit", "Delete", "Back"])  SWITCH action  CASE "Create":  data ← SHOW\_ANNOTATION\_MODAL(empty)  new\_id ← HTTP\_POST("/save", token, data)  HTTP\_POST("/annotations/" + new\_id + "/photos", token, data.photos)  CASE "Edit":  id, props ← PROMPT\_SELECT\_ANNOTATION(annotations)  data ← SHOW\_ANNOTATION\_MODAL(props)  HTTP\_PUT("/annotations/" + id, token, data)  HTTP\_POST("/annotations/" + id + "/photos", token, data.photos)  CASE "Delete":  id ← PROMPT\_SELECT\_ANNOTATION\_ID(annotations)  HTTP\_DELETE("/annotations/" + id, token)  CASE "Back":  BREAK LOOP  END SWITCH  END WHILE  END FUNCTION  // 4. Manage master data (Provinces / Cities / Place Types)  FUNCTION MANAGE\_PROVINCES(token)  CALL GENERIC\_CRUD("/provinces", token)  END FUNCTION  FUNCTION MANAGE\_CITIES(token)  CALL GENERIC\_CRUD("/cities", token, foreignKey="/provinces")  END FUNCTION  FUNCTION MANAGE\_PLACE\_TYPES(token)  CALL GENERIC\_CRUD("/place\_types", token)  END FUNCTION  // 5. Manage Users  FUNCTION MANAGE\_USERS(token)  CALL GENERIC\_CRUD("/users", token)  END FUNCTION  // Generic CRUD helper  FUNCTION GENERIC\_CRUD(endpoint, token, OPTIONAL foreignKey)  WHILE TRUE  records ← HTTP\_GET(endpoint, token)  DISPLAY\_TABLE(records)  action ← PROMPT\_TABLE\_ACTION(["Create", "Edit", "Delete", "Back"])  SWITCH action  CASE "Create":  data ← PROMPT\_INPUT\_FIELDS(records.schema, foreignKey)  HTTP\_POST(endpoint, token, data)  CASE "Edit":  id, props ← PROMPT\_SELECT\_RECORD(records)  data ← PROMPT\_INPUT\_FIELDS(records.schema, foreignKey, prefill=props)  HTTP\_PUT(endpoint + "/" + id, token, data)  CASE "Delete":  id ← PROMPT\_SELECT\_RECORD\_ID(records)  HTTP\_DELETE(endpoint + "/" + id, token)  CASE "Back":  BREAK LOOP  END SWITCH  END WHILE  END FUNCTION |

Gambar 7. Algoritma dari aplikasi PUNA-NPK

Algoritma SINEMA dimulai ketika pengguna membuka aplikasi dan melewati proses autentikasi—login atau pendaftaran—di mana kredensial disimpan dalam token JWT untuk sesi selanjutnya. Aplikasi akan membaca kredensial yang dimasukkan—username dan password—lalu mengirimkannya ke endpoint /login. Jika validasi berhasil, server mengembalikan token JWT yang disimpan di localStorage. Jika gagal, pengguna diminta mengulangi proses login hingga berhasil. Setelah itu, pengguna diarahkan ke halaman Dashboard.

Di Dashboard, aplikasi secara otomatis memanggil beberapa endpoint: /annotations, /provinces, /cities, /place\_types, dan /users. Dari sana dihitung jumlah setiap entitas untuk ditampilkan dalam kartu ringkasan (“cards”). Kemudian data anotasi dikelompokkan berdasarkan provinsi dan tipe lokasi untuk membuat grafik bar dan pie via Chart.js. Terakhir, aplikasi mengambil poligon anotasi terbaru dan menampilkannya dalam mini-map Leaflet, di mana peta difokuskan pada area tersebut. Setelah seluruh elemen visual ter-render, pengguna dapat memilih menu untuk melanjutkan ke fitur lain.

Jika pengguna memilih Manage Map, aplikasi melakukan fetch ulang /annotations dan menampilkan semua poligon di layer drawnItems Leaflet serta di tabel HTML. Saat klik “Create”, Leaflet.draw mengeluarkan event CREATED—saat itulah aplikasi membuka modal form terpadu yang sudah terisi default (tanggal hari ini) dan daftar dropdown provinsi/kota/tipe diisi via request ke API master data. Setelah pengguna mengisi semua atribut dan (opsional) memilih foto, maka form dikirim ke /save dan /annotations/:id/photos. Server menyimpan geometri dan metadata di PostGIS, serta file gambar di folder static/uploads.

Klik “Edit” di baris tabel memicu fungsi startEdit(id, props) yang memuat ulang dropdown dan field modal dengan data eksisting, lalu user dapat memperbarui teks atau menambah foto baru. Setelah submit, aplikasi memanggil PUT /annotations/:id dan mengunggah foto ke /annotations/:id/photos. Untuk “Delete”, aplikasi langsung memanggil DELETE /annotations/:id, menghapus catatan beserta relasi foto, lalu refresh peta dan tabel. Setiap perubahan segera di-refetch agar UI selalu sinkron dengan data terkini.

Fitur Manage Provinces, Manage Cities, Manage Place Types, dan Manage Users bekerja dengan skema CRUD serupa: aplikasi memanggil endpoint GET untuk memuat data, menampilkan tabel, lalu user dapat menambah, ubah, atau hapus baris dengan prompt (atau modal) sederhana. Semua permintaan dilengkapi header Authorization: Bearer <token>, sehingga hanya user terotentikasi yang berhak memodifikasi. Alur ini memastikan sinergi antara UI Leaflet, form modal, dan API RESTful di belakangnya berjalan terkoordinasi sesuai desain algoritmik.

A white sheet of paper with black text

AI-generated content may be incorrect.

Gambar 8. Sequence diagram aplikasi SINEMA

**5 Cara Penggunaan Aplikasi**

Berikut panduan langkah demi langkah untuk menggunakan SINEMA:

5.1. Login

1. Buka browser ke alamat aplikasi (misal http://localhost:5000).
2. Muncul halaman login:
3. Masukkan Username dan Password, lalu klik Login.
4. Jika kredensial benar, Anda akan diarahkan ke Dashboard.

5.2. Dashboard

1. Di Dashboard Anda akan melihat:
   * Cards: jumlah Anotasi, Provinsi, Kota, Tipe Lokasi, dan Users.
   * Mini-map: menampilkan poligon anotasi terbaru.
   * Grafik: distribusi anotasi per provinsi (bar chart) dan per tipe lokasi (pie chart).
2. Gunakan navbar di atas untuk pindah ke modul lain.

5.3. Manage Map (Pemetaan Longsor)

1. Pilih Manage Map dari navbar → tampil peta dan tabel:
2. Menambah Anotasi
   * Klik tombol Draw Polygon (ikon pensil).
   * Gambar poligon area longsor di peta.
   * Modal input terbuka, isi:
     + Provinsi, Kota, Tipe Lokasi (dropdown)
     + Lokasi kejadian, Tanggal (YYYY-MM-DD)
     + Kondisi area, Kondisi longsor, Dampak, Faktor penyebab, Mekanisme
     + Upload foto (bisa banyak)
   * Klik Save → data dan foto tersimpan, peta/tabel ter-refresh.
3. Mengedit Anotasi
   * Klik Edit pada baris tabel → modal terisi data eksisting + foto thumbnail.
   * Ubah nilai yang diinginkan, tambah foto baru bila perlu.
   * Klik Save → server perbarui data, refresh tampilan.
4. Menghapus Anotasi
   * Klik Delete pada baris tabel → konfirmasi → data & foto terhapus.

5.4. Manage Provinces

1. Klik Provinces di navbar → tampil halaman:
2. Menambah: isi nama provinsi → klik Add Province.
3. Edit/Delete: gunakan tombol pada tabel.

5.5. Manage Cities

1. Klik Cities → tampil dropdown Provinsi dan tabel Kota:
2. Menambah: pilih provinsi → ketik nama kota → klik Add City.
3. Edit/Delete via tombol pada tabel.

5.6. Manage Place Types

1. Klik Place Types → tampil halaman:
2. Menambah: isi nama tipe → klik Add Type.
3. Edit/Delete via tombol pada tabel.

5.7. Manage Users

1. Klik Users → tampil form registrasi dan tabel:
2. Menambah: isi username & password → klik Create.
3. Edit/Delete via tombol pada tabel.

**6. Diagram Relasi Database**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

Gambar 10. Diagram relasi database dari aplikasi SINEMA

Pada ERD SINEMA terdapat lima entitas utama dan satu entitas penunjang (foto), dengan relasi sebagai berikut:

1. Provinces
   * Atribut: id (UUID, PK), name (nama provinsi).
   * Relasi: satu provinsi dapat memiliki banyak kota (Provinces 1–∞ Cities) dan banyak anotasi longsor (Provinces 1–∞ Annotations).
2. Cities
   * Atribut: id (UUID, PK), province\_id (FK ke Provinces.id), name (nama kota).
   * Relasi: setiap kota milik tepat satu provinsi, dan satu kota dapat muncul di banyak anotasi (Cities 1–∞ Annotations).
3. Place\_Types
   * Atribut: id (UUID, PK), name (kategori lokasi, misal “Perbukitan”, “Pinggir Sungai”).
   * Relasi: satu tipe lokasi mengkategorikan banyak anotasi (Place\_Types 1–∞ Annotations).
4. Annotations
   * Atribut:
     + id (UUID, PK)
     + province\_id, city\_id, place\_type\_id (masing-masing FK ke tabel master)
     + occurrence\_location, occurrence\_date, area\_condition, landslide\_condition, landslide\_impact, causative\_factor, mechanism (detail kejadian)
     + geom (kolom GEOMETRY POLYGON di PostGIS)
   * Relasi:
     + Berada di satu provinsi dan satu kota
     + Termasuk satu tipe lokasi
     + Dapat memiliki banyak foto (Annotations 1–∞ Annotation\_Photos).
5. Annotation\_Photos
   * Atribut: id (UUID, PK), annotation\_id (FK ke Annotations.id), filename (nama file di static/uploads).
   * Relasi: setiap foto terasosiasi dengan tepat satu anotasi, dan satu anotasi dapat menampung banyak foto.

**7. Panduan Instalasi Aplikasi**

**7.1. Prasyarat Sistem**

* **OS**: Linux (Ubuntu 18.04+/20.04+), macOS, atau Windows
* **Python**: v3.8 ke atas
* **PostgreSQL**: v13+ dengan ekstensi **PostGIS** v3+
* **Git**: untuk clone code
* **pip** (bawaan Python)

**7.2. Clone Repository**

git clone https://git.brin.go.id/your‐group/sinema.git cd sinema

**7.3. Virtual Environment & Instalasi Dependencies**

# Buat & aktifkan venv python3 -m venv venv source venv/bin/activate # Linux/macOS # venv\Scripts\activate # Windows # Pasang paket Python pip install --upgrade pip pip install -r requirements.txt

**Catatan**: requirements.txt minimal memuat:

php

CopyEdit

Flask

Flask-SQLAlchemy

GeoAlchemy2

psycopg2-binary

shapely

Flask-JWT-Extended

**7.4. Konfigurasi Database**

1. **Jalankan Postgres & aktifkan PostGIS**

CREATE DATABASE sinema\_db; \c sinema\_db CREATE EXTENSION postgis;

1. **Atur koneksi di** app.py:

app.config['SQLALCHEMY\_DATABASE\_URI'] = \ 'postgresql://<user>:<password>@localhost/sinema\_db'

1. (Opsional) atur JWT\_SECRET\_KEY juga di app.py atau via env var:

export JWT\_SECRET\_KEY="ganti\_dengan\_rahasia\_Anda"

**7.5. Inisialisasi Skema & Tabel**

Di direktori proyek, dengan venv aktif:

python >>> from app import db >>> db.create\_all() >>> exit()

Ini akan membuat tabel users, provinces, cities, place\_types, annotations, annotation\_photos, dsb.

**7.6. Buat Folder Upload**

mkdir -p static/uploads chmod 777 static/uploads # agar Flask dapat tulis

**7.7. Menjalankan Server**

**a) Mode Development**

export FLASK\_APP=app.py export FLASK\_ENV=development flask run

**b) Langsung dengan Python**

python app.py

Server akan berjalan di http://127.0.0.1:5000/.

**7.8. Akses Aplikasi**

1. Buka browser ke http://127.0.0.1:5000/
2. Daftar user baru di /register, lalu login
3. Eksplore menu **Dashboard**, **Manage Map**, **Master Data**, **Users**

**8. Source code**

<https://git.brin.go.id/tauf022/sinema>