Agile Sprints - Backend Portfolio Project (MVP)

MVP Goal

A secure, backend-driven portfolio site where only you (the admin) can:

- Log in securely

- Add, update, and delete project entries

- View messages from visitors

Sprint 1: Project Setup & Config

Goal: Set up your environment, structure, secrets, and logging

- Initialize Git repo and virtual environment

- Install Flask, python-dotenv, bcrypt

- Create .gitignore, requirements.txt

- Create .env file to store secrets

- Create config.py to load settings from .env

- Create logging_config.py to configure rotating logs

- Set up app.py to load config, logging, and Flask app instance

Tests: None yet

Sprint 2: Admin Authentication System

Goal: Add secure login/logout functionality just for the admin

- Create utils/generate_hash.py to generate password hashes

- Create auth_controller.py for login, logout routes

- Store admin hash in .env, validate login with bcrypt.checkpw()

- Use Flask sessions to track admin login state

- Protect admin-only routes using session checks

- Create basic login HTML form (view)

- Add logging for login attempts and outcomes

Tests:

- Valid login sets session

- Invalid password rejected

- Logout clears session

Sprint 3: Project Management (CRUD for Portfolio)

Goal: Allow admin to manage portfolio projects

- Create project.py model class (OOP) to handle DB operations

- Create projects_controller.py to add, update, delete, list projects

- Admin can add a project (title, description, link, tags)

- Admin can update or delete projects

- Public users can only view project list

- Create Jinja templates for admin dashboard and project forms

Tests:

- Add project appears in DB

- Update project changes saved

- Delete project removed from DB

- Only admin sees edit/delete buttons

Sprint 4: Contact Form System

Goal: Visitors can send you a message from the site

- Create contact.py model class (name, email, message)
- Create contacts_controller.py for submitting + listing messages
- Public form sends POST to /contact
- Admin-only dashboard route to view messages
- Add input validation (empty fields, length checks)

Tests:

- Valid form stores message
- Empty form rejected
- Admin can view all messages

Sprint 5: Logging, Testing, and Final Polishing

Goal: Final touches before MVP is complete

- Add logging to all critical actions (login, CRUD, contact)
- Write unit tests for all models and controllers
- Add 404 error handler and custom error pages
- Review all routes for session protection
- Final test: walkthrough as admin and visitor

Tests:

- High test coverage (auth, project, contact logic)
- Logs written for key events
- Session-protected routes are not publicly accessible

MVP Completion

What You Will Have:

- A clean, OOP and MVC Flask backend

- Admin-only login with password hashing

- CRUD operations for showcasing your projects

- A working contact form with message storage

- Logging, security, config management using .env

- All code tested and pushed to GitHub