■ Security Lab - Feature Location Map

Generated: January 21, 2026 at 03:51 AM

■ Quick Navigation

■ Authentication & User Management

1. User Registration

Frontend	index.html, pages/register.html
Backend	backend/app.py (POST /api/register)
Functions	registerUser(), registerUserAPI()
Files	assets/js/script.js (line ~970) assets/js/api-client.js (line ~45)

2. User Login

Frontend	index.html
Backend	backend/app.py (POST /api/login)
Functions	loginUser(), loginUserAPI()
Files	assets/js/script.js (line ~1060) assets/js/api-client.js (line ~53)

3. User Dashboard Display

Frontend	pages/dashboard.html
Backend	backend/app.py (GET /api/users)
Functions	loadDashboard(), renderUserTable()
Files	assets/js/script.js (line ~707, 1040)

■ Security Testing

1. Dictionary Attack Simulator

Frontend	pages/security-testing.html
Backend	Client-side only (no backend call)
Functions	startCracking(), stopCracking(), resetCracking()
Files	pages/security-testing.html (line ~211-260) CryptoJS for hashing

2. Timing Attack Demonstration

Frontend	pages/security-testing.html
Backend	Client-side only
Functions	testTiming()
Files	pages/security-testing.html (line ~273-293)

3. Custom Wordlist Manager

Frontend	pages/security-testing.html
Backend	Python helper: scripts/attack_toolkit.py
Functions	loadWordlist(), generate_sample_wordlist()
Files	pages/security-testing.html (line ~296-310) scripts/attack_toolkit.py (line ~98)

■ Hash & Password Tools

1. Hash Calculator

Frontend	pages/hash-tools.html
Backend	Client-side only
Functions	calculateHash()
Files	pages/hash-tools.html CryptoJS library

2. Password Strength Analyzer

Frontend	pages/breach.html
Backend	backend/app.py (optional API)
Functions	analyzeStrength(), calculateBreachTime()
Files	pages/breach.html assets/js/script.js

3. Hash Export (Hashcat Format)

Frontend	pages/dashboard.html, pages/breach.html
Backend	backend/app.py + Python script
Functions	exportDatabaseForHashcat(), export_hashes_for_hashcat()
Files	assets/js/script.js (line ~372) scripts/attack_toolkit.py (line ~16)

■ Data Management

1. Export Database as JSON

Frontend	All pages (sidebar action)
Backend	localStorage fallback
Functions	exportDatabase()
Files	pages/*.html (all pages) assets/js/script.js

2. Clear Database

Frontend	All pages (sidebar action)
Backend	localStorage clear
Functions	clearData()
Files	assets/js/script.js localStorage.clear()

■ Backend API Endpoints

1. POST /api/register

Purpose	Register new user with password
Parameters	username, password, algorithm
Response	User ID, message
File	backend/app.py

2. POST /api/login

Purpose	Authenticate user
Parameters	username, password
Response	Success flag, message
File	backend/app.py

3. GET /api/users

Purpose	Fetch all users (for demo)
Parameters	None
Response	User list with usernames and algorithms
File	backend/app.py

4. GET /api/test

Purpose	Health check / test endpoint
Parameters	None
Response	Status message
File	backend/app.py

■ Python Scripts & Tools

1. Attack Toolkit

File	scripts/attack_toolkit.py
Purpose	CLI tool for hash export, wordlist gen, stats
Functions	export_hashes_for_hashcat(), generate_sample_wordlist(), show_crack_statistics()
Usage	python scripts/attack_toolkit.py

2. Documentation Generator

File	generate_documentation_pdf.py
Purpose	Generate PDF documentation
Functions	Main execution creates PDF
Usage	python generate_documentation_pdf.py

3. Startup Helper

File	scripts/START_APP.bat
Purpose	Initialize DB, install deps, start servers
Functions	Batch script automation
Usage	START_APP.bat

■ Core Assets

1. Main JavaScript Logic

File	assets/js/script.js
Purpose	User registration, login, dashboard, hashing, strength analysis
Key Functions	~1200+ lines of hybrid backend/localStorage logic

2. API Client

File	assets/js/api-client.js
Purpose	Fetch wrapper, registerUserAPI, loginUserAPI
Key Functions	~80 lines of API helpers

3. Main Styles

File	assets/css/style.css
Purpose	Global styling for all pages

4. Navigation Styles

File	assets/css/nav-styles.css
Purpose	Persistent sidebar navigation styling

■ Frontend Pages

■ Testing & Diagnostics

1. Feature Checker

File	feature-checker.html
Purpose	Verify all JS functions and DOM elements load correctly

2. Dashboard Debug

File	dashboard-debug.html
Purpose	Test dashboard functionality and API connectivity

3. Backend Connection Test

File	test-backend.html
Purpose	Verify backend endpoints and database connectivity

4. System Diagnostics

File	diagnostics.html
Purpose	Full system health check and feature validation

■ Database

1. SQLite Database

File	backend/instance/auth_security_lab.db
Purpose	Persistent user storage with hashed passwords
Tables	users (id, username, password_hash, algorithm, salt, created_at)

2. SQLAlchemy Models

File	backend/models.py
Purpose	ORM layer for database operations
Model	User class with fields (id, username, password_hash, algorithm, salt)

■ Project Summary

Component	Count	Location
Frontend Pages	8	pages/ folder + index.html
CSS Files	2	assets/css/
JavaScript Files	2 core + tests	assets/js/
Python Scripts	3+	scripts/ + root
Backend Endpoints	4+	backend/app.py
Database Tables	1 (users)	backend/instance/
Diagnostic Pages	4	Root directory

■ Key Insights

- **Hybrid Architecture:** Client-side logic with backend fallback (localStorage)
- **Security First:** Password hashing with MD5, BCrypt, and Argon2 algorithms
- **Educational Design:** Intentional vulnerabilities for learning (timing attacks, simple dict attacks)
- **Full Stack:** Flask backend, SQLAlchemy models, Bootstrap 5.3 frontend
- **Extensible Tools:** Python attack toolkit for offline testing and wordlist generation
- **Responsive UI:** Persistent sidebar navigation across all pages with dark theme
- **Testing Coverage:** Diagnostic pages for feature validation and debugging

■ How to Use This Map

1. Use the **Quick Navigation** table to find the category you're interested in
2. Navigate to the specified page to see detailed feature information
3. Check the **Files** or **File** column to locate the actual source code
4. For frontend features, look in **pages/** for HTML and **assets/** for CSS/JS
5. For backend features, check **backend/app.py** for API endpoints
6. For helper scripts, see the **scripts/** directory
7. Use diagnostic pages (feature-checker.html, etc.) to test features