SceneSolver – Software Development Life Cycle (SDLC)

1. Planning & Requirement Analysis

- Objectives:
  Automate crime scene analysis using AI, focusing on web-based workflows for rapid, accurate evidence extraction and scene classification
- Stakeholder Engagement:
  Collaborate with law enforcement, forensic teams, and developers to define needs and expectations.
- Security & Privacy:
  Assess data privacy, compliance requirements, and potential risks from the outset
- Deliverables:
  Project scope, timeline, feature list, and security requirements.

2. System Design

- Architecture:
  - Frontend: React.js (dark theme, red accents, animated backgrounds for modern UX)
  - Backend: Node.js (API logic), Flask (AI endpoints)
  - Database: MongoDB (cloud/local, scalable NoSQL storage)
- API Endpoints:
  `/upload`, `/analyze`, `/results`, `/user`, `/login`, `/signup`, `/dashboard`
- Security:
  - Threat modeling for data flows and endpoints.
  - Secure design patterns and access controls.
- Deliverables:
  System architecture diagrams, API specifications, and security documentation.

3. Implementation

- Frontend:
  - Upload page (validates 1 video or 4–15 images, drag-and-drop)
  - User page (fetches user details from backend)
  - Results tab (displays analysis results)
  - Header with navigation and authentication checks

        - Routing via react-router-dom, protected routes for authenticated users
        - Styling: dark mode, red accents, animated backgrounds, semi-transparent cards
- Backend:
    - Express.js routes, Flask AI service integration
    - JWT-based authentication and authorization
    - Input validation and error handling
- AI Models:
    - Fine-tuned CLIP for crime classification
    - Vision Transformer (ViT) for evidence detection
- Security:
    - Secure coding, code reviews, static code analysis

4. Testing

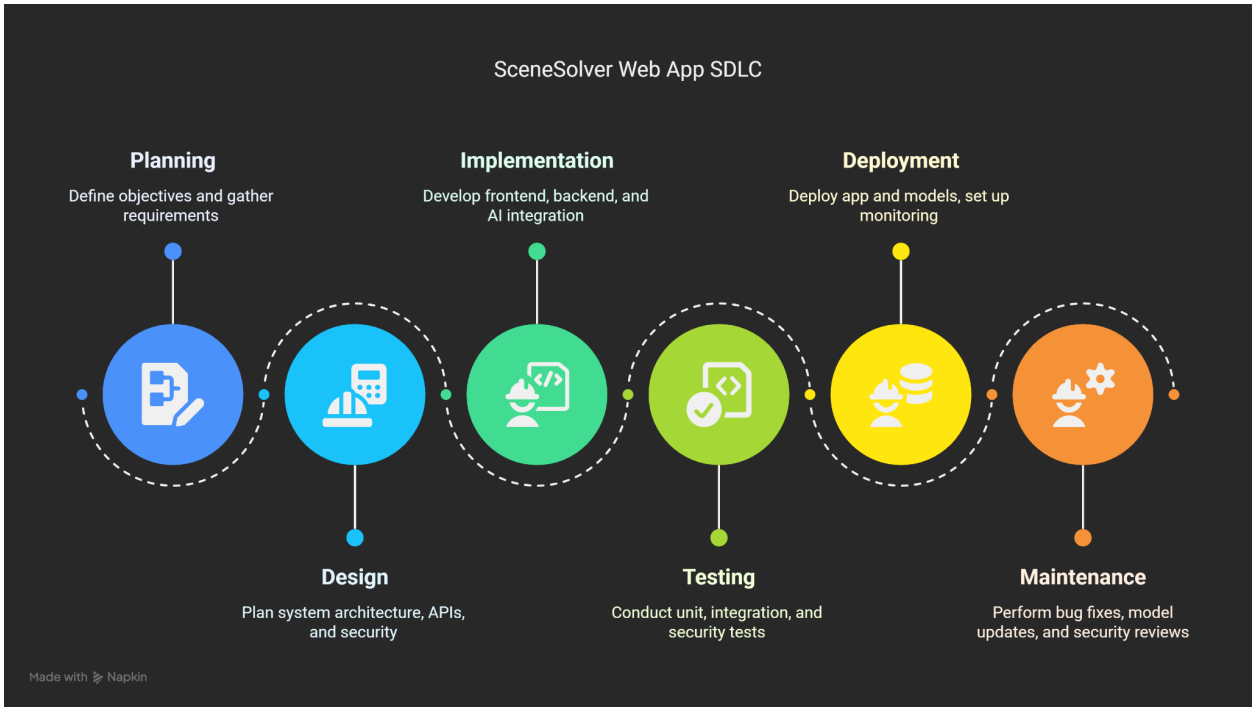
- Unit Testing:
    - Frontend components, backend APIs, AI model outputs
- Integration Testing:
    - End-to-end flows (upload → analysis → result)
- Security Testing:
    - Static/dynamic analysis (SAST/DAST), penetration testing, input validation, access control
- Performance & Usability:
    - Batch processing, offline mode validation
- Deliverables:
  Test cases, reports, and vulnerability remediation logs

5. Deployment

- Infrastructure:
    - Deploy Flask AI models (Docker/EC2)
    - Host web app (AWS Elastic Beanstalk/EC2, Vercel for frontend)
    - MongoDB Atlas for cloud database
- Security:
    - Secure deployment, environment hardening, vulnerability scanning
- Monitoring:
    - Logging, monitoring, and alerting for uptime and security
- Deliverables:
  Deployment scripts, environment docs, monitoring dashboards

6. Maintenance & Updates

- Ongoing Tasks:
  - Monitor AI performance, retrain models as needed
  - Patch bugs, update dependencies, improve UI/UX
  - Regular security reviews, vulnerability patching, compliance checks
  - Analyze usage logs for system improvements
- Deliverables:
  Maintenance logs, release notes, security audit reports




SceneSolver Website Architecture
State Management
API
useState
useEffect
Backend Integration
Flask API
MongoDB
Upload Page
Validation
Drag-and-Drop
User Page
MongoDB
Flask API
SceneSolver Website
Results Tab
File Processing
Analysis Results
Header
Navigation Links
Authentication Checks
Styling
Animated Background
Semi-Transparent Cards
Routing
react-router-dom
Protected Routes
Models
AI
CLIP
ViT
Made with Napkin