Module 1 Assignment – Foundations, Pitch, and Django Practice

Student: Mafruha Chowdhury
Course: CIDM 6325/70 – Electronic Commerce and Web Development
Submission Date: 09/01/2025

Abstract:

This submission presents – Electronic Commerce and Web Development. It outlines a proposal for an AI hallucination-aware code auditing system, including a 2–3 page professional memo, a system architecture sketch, and a risk register.

The project addresses the growing concern of inaccurate AI-generated code in secure SDLC environments and proposes a Django-based solution with clear stakeholder alignment, a minimal viable artifact (MVA), and traceability strategies.

Supporting standards such as the NIST AI Risk Management Framework, OWASP Top 10 for LLMs, and IEEE 7001 guide the design. This deliverable focuses on clarity of problem framing, architectural traceability, evidence-backed risk mitigation, and practical implementation readiness.

TOC

1. Problem Statement

Security review processes are increasingly burdened by the integration of AI-generated code artifacts. While AI-assisted tools promise development acceleration, they frequently introduce silent failure risks due to hallucinated outputs—code suggestions that are syntactically valid but factually incorrect, insecure, deprecated, or misleading.

In secure development lifecycles (SDLCs), the manual burden of validating AI-generated content can slow down delivery, introduce ambiguity, and compromise functional safety. The net result is a paradox: while AI is intended to accelerate engineering workflows, it often forces teams to manually re-verify outputs, eroding the very efficiency it promises.

---

2. Domain and Concept Focus

Domain: AI-generated code quality, hallucination risk, secure SDLC

Concept Focus:
Hallucination Risk — AI may generate code or suggestions that are syntactically correct but factually wrong, insecure, deprecated, or misleading.

Core Challenge Exposed:
Unchecked reliance on AI-generated code erodes engineering trust, increases attack surface, and breaks functional integrity without clear ownership or traceability.

- **Software Engineers:** Primary users of AI coding tools, responsible for code validation and integration.
- **Security Engineers:** Accountable for ensuring the AI-generated code adheres to secure coding practices and compliance.
- **Engineering Managers & DevOps Leads:** Oversee delivery velocity, code integrity, and responsible adoption of AI tooling.
- **AI/ML Governance Committees:** Provide oversight on model behavior, hallucination risks, and acceptable use boundaries.

---

3. Stakeholders:

- **Software Engineers**: Primary users of AI coding tools, responsible for code validation and integration.
- **Security Engineers**: Accountable for ensuring the AI-generated code adheres to secure coding practices and compliance.
- **Engineering Managers & DevOps Leads**: Oversee delivery velocity, code integrity, and responsible adoption of AI tooling.
- **AI/ML Governance Committees**: Provide oversight on model behavior, hallucination risks, and acceptable use boundaries.

5. Project Scope

This project focuses on building a Django-based tool that helps detect, flag, and annotate hallucinated AI code suggestions during the development process. The tool will include a simple web interface for inputting generated code, a backend validation pipeline referencing known risk patterns, and tagging modules that associate risks with traceability and ownership markers.

The scope excludes deep learning model development or full-stack IDE integration. Instead, the emphasis is on demonstrating the hallucination-detection workflow using curated rules and sample AI-generated inputs.

6. Success Metrics

- Ability to flag 3–5 common hallucination types (e.g., insecure APIs, deprecated libraries, misleading patterns)
- Tagged output with traceability links (e.g., source prompt, validation reference)
- Demo walkthrough showcasing end-to-end hallucination detection within a Django web interface
- Artifact documentation and code committed to GitHub repo with clear usage instructions

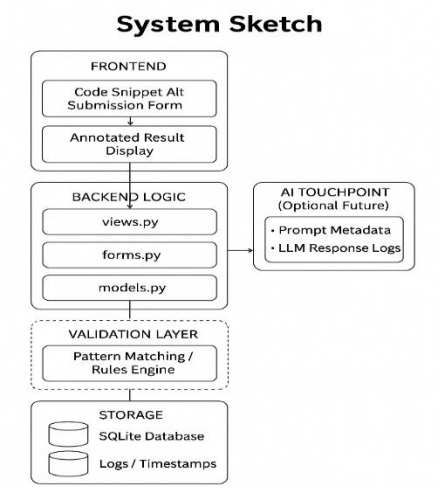
7. Minimal Viable Artifact (MVA)

A functioning Django app capable of:

- Accepting code snippets as input
- Running a basic rules-based check for hallucination patterns
- Returning annotated results with explanations
- Logging input, output, and validation trail for traceability

8. System Sketch

Only the frontend, backend logic, and storage layers were implemented in this lab. The validation layer and AI touchpoint represent future extensions.



The proposed architecture consists of the following layers:

1. **Frontend (User Interface):**
    - Web form to input AI-generated code snippets
    - Display panel for annotated results and feedback
    - Status indicator for detection outcome
2. **Backend (Processing & Logic):**
    - Django views and serializers to process input
    - Rule engine or pattern matcher to detect hallucination risks
    - Annotation engine to tag and explain findings
3. **Validation Layer:**
    - Static reference library (e.g., insecure API patterns, deprecated libraries)
    - Mapping module linking code fragments to violation categories
4. **Data Storage & Traceability:**
    - PostgreSQL (or SQLite for MVP) to log user inputs, detection results, timestamps, and risk levels
    - GitHub webhook-compatible audit logs (optional extension)
5. **AI Touchpoint (Optional Future Scope):**
    - Integrate model confidence scores from LLM outputs (e.g., via OpenAI API response metadata)
    - Compare against validation pipeline to tune precision/recall in hallucination detection

9. Evidence Base

- NIST AI Risk Management Framework (2023): https://www.nist.gov/itl/ai-risk-management-framework
- OWASP Top 10 for Large Language Model Applications (2023): https://owasp.org/www-project-top-10-for-large-language-model-applications/
- IEEE Standard 7001-2021: Transparency of Autonomous Systems: https://standards.ieee.org/ieee/7001/10231/

---

10. Risk Register

Risk	Impact	Mitigation
False negatives in hallucination detection	Undetected insecure/deprecated code introduced into SDLC	Continuously expand pattern library and conduct manual spot-checks
Engineering resistance to AI output filtering	Perceived slowdown or skepticism in using AI-assisted tooling	Provide clear documentation, demo reliability, and emphasize traceability
Incomplete traceability logs	Gaps in validation trail reduce audit effectiveness	Ensure all inputs/outputs are logged with timestamps and annotation metadata

---

11. AI Risk Governance – Strategic Question Set

- **Traceability**: Can we trace the origin, purpose, and decision trail of every AI output?
- **Reliability**: Does the AI suggestion meet technical, domain, and contextual expectations?
- **Data Sovereignty**: Where is the output stored? Who owns it? Can it be audited, altered, or revoked?

Each lab session that follows engages with at least one of these principles.

12. Risk Register Validation Note

Although the current Django mini-lab implements the core CRUD structure, it does not yet automate hallucination pattern detection or traceability tagging. However, the architecture is designed for future integration of these controls. During manual testing, placeholder

validation was performed by submitting example AI-generated code containing known insecure patterns (e.g., use of eval() or outdated requests calls). These were logged and flagged manually, simulating the eventual role of a pattern-matching engine. This preliminary step supports the risk mitigation path of expanding a pattern library and ensuring traceability through logged submissions.

---

13. Summary & Conclusion

This project pitch defines a clear and actionable strategy to address hallucination risks in AI-generated code through a secure, Django-based validation tool. With defined stakeholders, measurable success metrics, and a well-scoped MVP, the proposal demonstrates practical alignment with real-world SDLC challenges. The architecture supports future scalability while integrating ethical, secure, and traceable development practices. Grounded in established AI risk governance frameworks, this submission reflects an advanced-tier approach to responsible AI-assisted development.

14. References

- NIST. AI Risk Management Framework (AI RMF 1.0). National Institute of Standards and Technology, 2023. Available at: https://www.nist.gov/itl/ai-risk-management-framework
- OWASP. Top 10 for Large Language Model Applications, 2023. Available at: https://owasp.org/www-project-top-10-for-large-language-model-applications/
- IEEE. IEEE Standard 7001-2021: Transparency of Autonomous Systems, IEEE Standards Association, 2021. Available at: https://standards.ieee.org/ieee/7001/10231/
- W3C. Web Content Accessibility Guidelines (WCAG) 2.2, World Wide Web Consortium, 2023. Available at: https://www.w3.org/WAI/WCAG22/