PixelForge Nexus: Submission Guide

Your assignment requires two main deliverables: a written report (DOCX) and a video report. Both need to be linked from the DOCX file and hosted on Google Drive with specific sharing settings.

I. Written Report (.docx File)

This is your primary submission document. It must be a .docx file and will contain three main sections: the links, and the individual report brief.

A. Top Section: Required Links

At the very top of your .docx file, you must include two links:

1. Link to your Prototype Complete Source Code (Google Drive Folder):

    - What to include: The entire PixelForgeNexus project folder, containing all files (app.py, models.py, forms.py, config.py, extensions.py, templates/, static/, instance/, migrations/, venv/ - though venv can be omitted to save space, but ensure your requirements.txt is present if you omit venv).

- Commenting: Ensure your source code is "appropriately and correctly commented." Explain logic, security considerations, and any non-obvious parts.
- Assumptions: Clearly identify any assumptions you have made in your code or design (e.g., "Assumed production deployment would use HTTPS and a WSGI server," "MFA is a placeholder for this prototype").
- Google Drive Sharing: Crucially, the folder link must be set to "Anyone with the link can view." If it's not, your marker won't be able to access your code.

2. Link to your Video Report (Google Drive Video File):
    - What to include: The singular video file (MP4, MOV, etc.).
    - Google Drive Sharing: Crucially, the video link must also be set to "Anyone with the link can view."

B. Main Section: Individual Report Brief (Approx. 2000 words)

This is the core of your written submission, detailing your design, development, and testing. Structure it according to the grading rubric sections.

1. System Design (35% Weighting) * System Overview: Briefly describe "PixelForge Nexus" and its purpose. * Design and Security Principles: * Explain what design and security principles you considered (e.g., Least Privilege, Defense in Depth, Secure by Default, Minimizing Attack Surface, Separation of Concerns, Fail-Safe Defaults). * For each principle, describe how you applied it in your system's design (e.g., "Least Privilege was applied by defining distinct roles (Admin, Project Lead, Developer) with granular permissions..."). * Explain why these principles were chosen and how they enhance the system's functioning and security. * Threat Model: * Identify and prioritize potential security risks relevant to your system (e.g., Unauthorized Access, SQL Injection, XSS, Broken Access Control, Sensitive Data Exposure, Insecure Direct Object References). * For each threat, describe the specific mitigation strategies implemented in your design. * Access Control Mechanisms: Detail how role-based access control (RBAC) is implemented (e.g., using Flask-Login's @login_required and custom is_admin(), is_project_lead() checks in routes). Provide examples of which roles can access which functionalities. * Data Encryption Strategies: Discuss password hashing (bcrypt via Werkzeug) and storage. Mention any other data at rest or in transit considerations (e.g., "For a production system, document storage would be encrypted at rest, and all communication would be over HTTPS"). * Documentation: Ensure this section is detailed and well-justified, illustrating how security principles are applied.

2. Security Testing and Analysis (35% Weighting) * Evaluation of Security Techniques: * Describe the security techniques you applied (e.g., input validation, password hashing, secure session management, access control enforcement). * Critically evaluate their effectiveness. What are their strengths? What are their limitations in this prototype? * Issues Discovered and Proposed Solutions: * Describe any security issues you intentionally left out for simplification (e.g., "MFA was identified as a critical security enhancement but was not fully implemented in the prototype due to scope/time; a proposed solution would involve integrating a TOTP library like

pyotp"). * If you performed any manual testing (e.g., trying to access admin pages as a developer, submitting malicious input), describe your findings and how your implemented controls mitigated them. * If you used any "code scanning reports" (e.g., if you ran a linter with security checks, or a basic static analysis tool), include relevant findings and how you addressed them. * Test Cases and Results: Provide a summary of test cases related to security (e.g., "Attempted login with invalid credentials," "Attempted to add project as a developer," "Attempted to upload non-document file types"). Show the expected vs. actual results. * Mitigation: Explain how your proposed solutions (or already implemented ones) can mitigate the identified problems.

3. System Development (20% Weighting) * Development Methodology & Techniques: * Explain the methods and techniques you followed (e.g., Agile-like iterative development, use of Flask, SQLAlchemy, WTForms, Flask-Login). * Discuss the stages of your development life-cycle (e.g., specification/understanding requirements, design based on security principles, iterative development/ coding, testing, deployment considerations). * Functional Prototype Demonstration: * Describe how your prototype complies with the design (e.g., "The implemented login system utilizes bcrypt for password hashing as designed," "Role-based access control is enforced at the route level"). * Explain the proper functioning of each core feature (Project Management, Team Assignment, Asset & Resource Management) and how security mechanisms are integrated into their operation. * Secure Coding Practices: * Detail the secure coding practices you adhered to (e.g., using an ORM to prevent SQL injection, proper input validation, using Flask's built-in templating for XSS prevention, secure session handling via Flask-Login). * Provide examples from your code where these practices are evident. * Legal and Ethical Context: Briefly discuss any legal/ethical considerations relevant to a system handling user data and project documents (e.g., data privacy, access control, responsible data handling).

4. Formal Methods (10% Weighting) * Behavioral Model: * This section requires you to apply formal methods. You could use a state machine, a sequence diagram, or a simple formal specification language (e.g., using pseudocode or a simplified notation) to model a key behavior of your system, particularly one with security implications (e.g., the login process, or the process of assigning a user to a project). * Describe the states, transitions, and conditions. * Verification of Correctness: * Based on your behavioral model, describe how you would verify its correctness with respect to its security specification. * For a prototype, this might be a conceptual discussion rather than a full formal proof. For example, "A formal verification of the login state machine would ensure that only successful authentication leads to the 'logged in' state, and all other inputs lead to an error state without granting access."

II. Video Report (8 minutes or less)

This video is your chance to visually demonstrate your working prototype and explain its key aspects.

- Duration: Keep it concise, 8 minutes or less.
- Content (Fly-through):
  - Introduction: Briefly state your name and the project name ("PixelForge Nexus").
  - Login System: Demonstrate the login process with the admin user. Show both successful and (briefly) failed login attempts to highlight the feedback.
  - Role-Based Access Control (RBAC):
    -

Log in as admin: Show access to "Add Project," "Register User," "Manage Users." Demonstrate adding a project and registering new users (e.g., a Project Lead, a Developer).

  - Log in as a Project Lead: Show access to projects they lead (if any), and options to "Assign Team Members" and "Upload Documents" for their projects. Show that they cannot access Admin-only features like "Register User."

  - Log in as a Developer: Show access to projects they are assigned to. Demonstrate viewing project details and documents. Show that they cannot access Admin or Project Lead specific features.

- Project Management: Demonstrate adding a project (as admin), viewing projects (all roles), and marking a project as "Completed" (as admin).

- Team Assignment: Show a Project Lead assigning a developer to their project.

- Asset & Resource Management: Demonstrate uploading a document to a project (as Admin/Project Lead) and then viewing that document (as any assigned user/lead/admin).

  - Account Settings: Briefly show the "Update Password" form.
  - Security Mechanisms Highlight: As you demonstrate features, use voice-overs/text overlays to point out how security is implemented (e.g., "Here, the system is enforcing role-based access control, preventing this developer from accessing admin functions," "Password hashing is handled securely on the backend," "Input validation prevents malicious data entry.").
  - Development Process/Testing (if applicable): You can briefly mention or show aspects of your development environment or a quick test case if it helps illustrate a point from your report.
- Presentation:
  - Voice-overs: Explain what you are doing and why.
  - Text Overlays: Use text overlays for key terms, highlighting specific features, or providing brief explanations.
  -

Clarity: Ensure your screen is clear and easy to see.

- Submission: Upload the video to Google Drive and ensure the link is set to "Anyone with the link can view."

III. General Tips for Success

- Start Early: This is a comprehensive assignment. Plan your time for coding, testing, writing the report, and recording the video.
- Cross-Reference: Ensure your written report and video complement each other. The video demonstrates, the report explains in detail.
- Be Specific: When discussing security principles or testing, provide concrete examples from your code or demonstrations.
- Proofread: Carefully proofread your written report for grammar, spelling, and clarity.

- Test Links: Before submission, test both Google Drive links (source code folder and video) from a different browser or incognito window to ensure they are publicly accessible.

This detailed guide should help you structure your submission effectively. Good luck!