**Code Review Summary**

**Overview**

Throughout my academic career and culminating in this capstone course, I have developed and continuously improved three significant software projects: the CS 330 3D Desk Scene, the CS 340 MongoDB Dashboard, and the CS 305 Secure File Transmission system. Each project presented unique challenges and learning opportunities, allowing me to apply foundational concepts and progressively enhance functionality, security, and user experience. The following review outlines the evolution of each project’s codebase, the technical improvements made, and how these refinements demonstrate my growing expertise as a software engineer.

**CS 330 3D Desk Scene**

* **Original Implementation:**  
   The original codebase consisted of a basic 3D scene constructed with C++ and OpenGL, featuring a desk with several objects. Core components such as SceneManager and ShaderManager were designed to organize rendering and shader logic. Initial lighting was simple, relying on basic ambient and directional lights, and texture mapping was minimal, with few surfaces textured for realism.
* **Enhancements and Technical Details:**  
   To elevate the visual fidelity and user engagement, I enhanced the lighting model by implementing Phong shading, which calculates ambient, diffuse, and specular components per pixel, providing realistic reflections and dynamic shadows. This required rewriting and optimizing GLSL shader programs to handle multiple light sources and object materials dynamically.

Texturing was improved by integrating high-resolution image maps and applying texture coordinate transformations to correctly map textures onto irregular objects, minimizing distortion and seams. Additionally, I incorporated normal mapping techniques to simulate fine surface details without increasing geometric complexity.

For interactivity, I developed an input handling system enabling users to manipulate the camera angle, zoom level, and selected objects in real time. This involved capturing keyboard and mouse events, updating transformation matrices, and synchronizing these changes smoothly with the rendering loop to maintain high frame rates.

* **Challenges:**  
   Managing GPU shader uniform variables efficiently was a technical hurdle, requiring careful synchronization between CPU-side state and GPU-side shader inputs. Handling user input in a non-blocking, event-driven manner to prevent frame drops demanded an understanding of both OpenGL and Windows event processing.
* **Outcome:**  
  These improvements transformed the project from a static demonstration into an interactive, photorealistic 3D environment. The enhancements exhibit advanced graphics programming skills, mastery of shader pipelines, and user interface integration—core competencies in software engineering and real-time visualization.

**CS 340 MongoDB Dashboard**

* **Original Implementation:**  
   Initially, the project consisted of Python scripts implementing CRUD operations against a MongoDB database containing animal shelter data. While functional, this command-line interface lacked interactivity and visualization features necessary for end-users to efficiently analyze and manage the data.
* **Enhancements and Technical Details:**  
   Building on the backend foundation, I designed and implemented a **web-based dashboard** using the Dash framework for frontend visualization and Flask for backend routing. A key enhancement was the integration of **MongoDB Change Streams**, allowing the dashboard to reflect real-time database changes without manual refresh, achieved via asynchronous callbacks and WebSocket communication.

Advanced filtering mechanisms were added, enabling users to query the dataset based on species, age, adoption status, and other attributes, with results immediately reflected in dynamic tables and charts powered by Plotly. Export functionality was also implemented, allowing users to download filtered datasets in CSV and PDF formats for offline reporting.

To decouple frontend and backend, I developed a RESTful API using Flask, facilitating scalable access to data and enabling future integration with other client applications or services.

* **Challenges:**  
  Implementing real-time updates required deep understanding of asynchronous programming and event-driven architectures to ensure that UI components remained responsive. Maintaining data consistency and security during concurrent database operations was another critical focus.
* **Outcome:**  
   The dashboard greatly improved user engagement by providing a seamless, responsive interface tailored to client needs. This work demonstrates proficiency in full-stack development, real-time data handling, API design, and client-focused software engineering practices.

**CS 305 Secure File Transmission**

* **Original Implementation:**  
   The project involved developing a secure file transmission system using Spring Boot. The baseline implementation supported file uploads/downloads over HTTPS, with data integrity checks performed via SHA-based hashing algorithms.
* **Enhancements and Technical Details:**  
   To strengthen security and performance, I integrated multiple cryptographic hash functions: **SHA-256, SHA-512, and BLAKE2, benchmarking each for speed and collision resistance. This benchmarking guided the dyn**amic selection of the optimal hashing algorithm based on file size and system load.

The system’s communication channel was secured using SSL/TLS protocols, with careful configuration of certificates and cipher suites to mitigate vulnerabilities such as downgrade attacks or weak encryption.

I designed and implemented a suite of automated security tests based on OWASP guidelines, targeting common vulnerabilities like injection flaws, improper authentication, and sensitive data exposure. Continuous integration pipelines were configured to run these tests upon each build, ensuring ongoing security compliance.

Furthermore, I enhanced the user experience by adding a performance monitoring dashboard that displays hashing speeds, encryption status, and transfer success rates in real time.

* **Challenges:**  
   Balancing cryptographic security strength with application performance required careful profiling and optimization. Implementing comprehensive security tests necessitated deep understanding of both the application logic and security standards.
* **Outcome:**  
   These enhancements resulted in a robust, secure file transmission system capable of protecting sensitive data against modern threats while providing transparency and usability for end users. This artifact reflects applied expertise in software security engineering, cryptography, and secure software lifecycle management.