**Software Systems Competency Assessment**

**1. Introduction to Software Systems**

In my MS-CISBA program, two pivotal courses enhanced my understanding and capability in building software systems with newer technologies:

* **CIDM 6330 – Software Engineering and Systems (Dr. Babb, Spring 2023)**
* **CIDM 6325 – Electronic Commerce and Web Development (Dr. Babb, Fall 2022)**

CIDM 6330 focused on software architecture patterns, test-driven development, domain modeling, and enterprise design using Python and Django. CIDM 6325 emphasized web technologies, Django-based site building, eCommerce components, user experience, and SEO strategies. As a practicing software engineer, these courses offered both familiar territory and valuable challenges - especially when stepping back to write formal test cases before writing functional code. This discipline deepened my architectural thinking and improved the robustness of my implementations.

**2. What I Know (Strengths)**

My core competencies in software systems include:

* **Web Application Development**: Using Django, I created several full-stack web apps. These included models, views, templates, and database-backed functionality.
* **Software Architecture Patterns**: From Martin and Percival’s texts, I gained fluency in layered architecture, service-based design, CQRS, and domain-driven design.
* **Test-First Development**: Writing unit tests before code (TDD) helped me think through edge cases and system behaviors before implementation.
* **Object-Oriented Design**: I applied encapsulation, modularization, and clean separation of concerns throughout backend implementations.
* **UX and Frontend Design**: My final CIDM 6325 project emphasized interface design principles, responsive layout, and integration of SEO metadata.
* **Version Control and Collaboration**: Throughout both courses, I maintained GitHub repositories, documented project milestones, and pushed clean, testable code.

**3. Where I Am Weak**

Despite my strengths, I found certain areas challenging:

* **Advanced Backend Integration**: Concepts such as messaging queues, event-driven patterns, and CQRS architecture required significant conceptual focus.
* **DevOps and Deployment**: While I successfully ran applications locally, I am still learning how to automate deployment to the cloud and integrate CI/CD pipelines.
* **Front-End Frameworks**: Although I can work with Django templates and basic CSS, frameworks like React or TailwindCSS remain outside my comfort zone.

**4. What I Wish I Knew**

To round out my software systems knowledge, I’d like to explore:

* **REST API Security and Testing**: Including token-based authentication and Postman-driven testing for full CRUD operations.
* **Security Best Practices**: Applying OWASP principles to prevent injection attacks, cross-site scripting, and enforce secure logins.

The test-driven approach in **CIDM 6330** was particularly eye-opening. Even as a working software engineer, being forced to plan tests before writing a single line of logic code made me a more disciplined and forward-thinking developer.

**5. Supporting Evidence**

Throughout both courses, I completed and submitted:

* **Python Architecture Project** (CIDM 6330): Developed a complete domain-driven system using layered architecture, unit testing, and modular design.
* **Web Application Project** (CIDM 6325): Built a Django-based web store with SEO elements, content management, and a functional UX layer.
* **Codebase Artifacts**: All project code was version-controlled on GitHub, including automated tests, models, and deployment scripts.

These artifacts are stored in a SS folder in my Capstone GitHub repository.

**6. Capstone Readiness and Integration**

Software systems knowledge is a cornerstone of my Capstone prototype. I designed and built a reusable, modular Jupyter/Colab notebook as the primary artifact. My notebook:

* Loads structured data into memory from external sources (GitHub CSV datasets)
* Allows the user to define personal effort preferences through the user\_profile object
* Applies an advisor scoring engine to recommend bank bonus offers based on user-defined criteria
* Produces visual outputs (tables and charts) to help users interpret results and make informed decisions
* Is structured following software engineering best practices to allow for future modular expansion

The Jupyter/Colab environment proved extremely valuable as an interactive prototyping platform. It allowed me to explain, test, document, and visualize the system all in one place while maintaining a logical, readable workflow for the end user.

Looking ahead, I recognize that a notebook-based system is not a long-term production solution. My next phase will involve evolving the codebase into a web or mobile application to improve usability and accessibility for a wider audience. This design philosophy of building a flexible prototype with future scalability in mind reflects the software systems concepts I learned throughout the MS-CISBA program.

This competency integrates with:

* **Data Management**, by providing the software logic to read, clean, and structure raw data for analysis
* **Data Analytics**, by offering an interface for algorithmic scoring and visualization of results
* **Cybersecurity**, by raising awareness of secure access, protecting private data inputs, and emphasizing clean, controlled data handling within the system

**7. Conclusion**

While I entered the program as a software engineer, these courses sharpened and matured my system thinking. Writing code within a framework of tests, architecture, and business purpose elevated how I approach even familiar development tasks. I feel equipped to build a professional-grade Capstone prototype and excited to apply what I’ve learned to future projects.