KIMPAY, Ralph Joshua T.
IT 313

The election system, when viewed through the lens of functional requirements and design, demonstrates the essential connection between well-defined system objectives and the structured development process outlined in the Software Development Life Cycle (SDLC). Each SDLC phase—planning, analysis, design, implementation, testing, and maintenance—plays a vital role in ensuring that the system operates with transparency, accuracy, and security, which are critical for election integrity. Clearly identifying functional requirements such as voter authentication, vote recording, result tallying, and data security ensures that every system feature aligns with real-world needs and legal standards.

From a system modeling perspective, the use of diagrams such as Data Flow Diagrams (DFD) or Unified Modeling Language (UML) provides a clear visualization of how data moves through the system. These models help both developers and stakeholders understand processes such as voter registration and result verification, promoting transparency and reducing ambiguity during implementation. Modeling also ensures that all inputs, processes, and outputs are logically structured and interdependent.

In terms of architectural design, selecting an appropriate architecture—such as a client-server or web-based architecture—is crucial for scalability, security, and performance. A modular design approach allows for future enhancements, like integrating biometric verification or blockchain-based vote recording.

Overall, the election system's success depends on the clarity of its functional requirements and how well its design adheres to SDLC principles. A well-modeled and architecturally sound system not only ensures functionality and reliability but also reinforces public trust in the electoral process. This reflection underscores the importance of disciplined system development practices in achieving both technical excellence and societal impact.