Reflect on and include a discussion of the following topics, using readings from throughout the course to support your views.

* Adoption of a secure coding standard, and not leaving security to the end

Adopting secure coding standards and following best practices would ensure that vulnerabilities and deficiencies in software are detected early on in the development process or lifecycle. This is the goal of the concept of “not leaving security to the end.” As a secure coding best practice, this concept means that security considerations should be integrated into every stage of the software development lifecycle SDLC, from initial design and development through testing and maintenance. Having security in mind would ensure the development of secured, functional and robust software or applications that would be able to stand the test of time. Using automation and unit testing to constantly test code bases for inherent logical errors, syntax or runtime errors would help developers fix any identified error and develop a rigid model that would be able to withstand attacks in the future. In other words, finding and resolving code errors early on allows developers to identify potential deficiencies that attackers might seek to exploit in the future. That way, they can resolve the issue and also proactively plan ahead having a strong incident response plan to counteract and neutralize security attacks. For instance, with the defense in depth (DiD) security architecture implemented for Green Pace, security attack is almost impossible because the system at Green Pace is almost impenetrable. The preexisting intricate security system has been improved with more automation and better incident response policy among other things. The new security policy ensures that all code defects are detected early and resolved, ensuring that the final product or application is secure and uncompromisable even with the most sophisticated attacks.

* Evaluation and assessment of risk and cost benefit of mitigation.

Evaluating and assessing the risk and cost-benefit of mitigating security vulnerabilities could be multifaceted. Firstly, identifying potential vulnerabilities through static analysis and code reviews is crucial. The risks associated with these vulnerabilities can vary from data breaches to system crashes, impacting both security and stability. Mitigating these issues often incurs significant costs, including developer time, potential re-architecting of code, and possible performance trade-offs. However, the benefits of addressing these vulnerabilities usually outweigh the costs, as they enhance overall system security, protect sensitive data, and maintain user trust. A thorough cost-benefit analysis should consider both the immediate expenses and the long-term benefits of a more secure and resilient codebase.

* Zero trust

Adhering to zero trust standard means that all hardware and personnel accessing network resources are constantly verified and validated on the network. This ensures that unauthorized users and hardware are denied system or network access helping to safeguard sensitive data and prevent data breach attacks and hijacks. Zero Trust is a security model based on the principle of “no one is safe” and "never trust, always verify." This model assumes that threats could be both external and internal to the network, so it continuously verifies every user and device attempting to access resources, regardless of whether they are inside or outside the network perimeter. Developers cannot trust anything entering or leaving the network. As such, it is critical that everything in the network is continuously verified to ensure that suspicious hardware, users or activity are detected and reported. This practice ensures that all warning signs and red flags of imminent attacks are noted so that a comprehensive plan of action could be put in place to stop them.

* Implementation and recommendations of security policies

Implementing security policies is crucial to mitigating vulnerabilities in code bases. It aims at ensuring application functionality and data security. Security policies implementation involves defining and enforcing rules to protect resources and ensure compliance with regulations. Recommendations for effective implementation include: a clear comprehensive documentation of policy, regular training, stringent access control policy, continuous monitoring and policy reviews and updates. Additionally, some key recommendations for effective security policies include data encryption policy (data in flight, in use and at rest), Triple A policy (authentication, authorization and accounting), automation, stringent access control policy, and effective incident response policy as seen in the security policy project created in this class.