# CS 405 Project Two Speaking Notes

Complete this template by replacing the bracketed text with the relevant information.

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Welcome to the Green Pace security policy presentation. My name is Jason Gionfriddo. Thank you for taking the time t review this. |
| **2** | Defense in Depth has many layers, but how many is determined by the amount of time and resources can be dedicated to not only implementing the layers but also updating and maintaining those layers to ensure optimal functionality. Green Pace Secure Development Policy will focus primarily on coding and testing standards in the app security and endpoint security layers. |
| **3** | Likely: vulnerabilities that are either easily missed, not explicitly checked for, or the vulnerability lies in edge cases. Priority vulnerabilities are those that present immediate risk to sensitive data or critical operating functionality. Low priority vulnerabilities are still important but do not pose immediate threats to sensitive data. These may include functions that are not optimized in terms of Big’O notation and can be resolved when immediate threats are at a minimum. Unlikely vulnerabilities are easily identified, for example logic flaws written by jr. devs. |
| **4** | The ten pricipals that are listed here are defined and talked about in further detail in the policy. For now, you should be aware that they have been hand picked to minimize threats to our data and operations. |
| **5** | The security policy also goes into these coding standards in great detail. They have been well defined with examples of acceptable practices, unacceptable practices, related policy references and tolls to help abide by these standards. |
| **6** | Encryption at rest   * Sensitive data such as but not limited to account numbers, card numbers, passwords, and all other information that must conform to regulatory restrictions will be hashed before storage with a minimum of 4 iterations by the hashing program. This should add a sufficient amount of protection of sensitive data in the event of a breach of system security   Encryption at flight   * When data that fits requirements mentioned in the encryption at rest section will be encrypted using TLS protocol. This will help protect sensitive data as it is being transferred and may be susceptible to hacking attacks   Encryption in use   * Encryption in use is the hardest to regulate and implement. The least amount of data necessary to perform business actions should be requested and viewable by those dealing with data, whether it be creating, reading, updating, or deleting data. |
| **7** | Authentication is the verification that a potential user is who they claim to be. Usernames, passwords, and multifactor verification with a minimum of 2 layers should be used to authenticate users  Authorization is the access that is to be granted to an authenticated system user. There will be at the minimum departmental separation of concerns, and the principal of the least amount of privilege given to complete business tasks will be strictly adhered to.  Accounting will log system transactions for review at a later date. These reviews will be performed by multiple authorized and qualified personnel with the intent of reporting to the adherence of security protocol and effectiveness of said protocols. |
| **8** | Unit test explanation |
| **9** | Unit test explanation |
| **10** | Unit test explanation |
| **11** | Unit test explanation |
| **12** | DevSecOps incorporates a security first mindset into SDLC as opposed to typical devops where infosec often comes at the end. DevSecOps take Continuous Integration/Continuous Deployment and includes security checks at every phase. The phases include Plan, Code, Test, Release, and deploy. In the planning phase, the how, where, and whens of testing will be determined. These tools will start to come into effect for the code, build, test, and release phases. |
| **13** | Ashley Madison example. Security measures can be added to the CI/CD pipeline, as discussed above. Each time a developer builds a code, he runs a CI/CD pipeline tool that does all the necessary processes, i.e., pushing code to a shared repository and sending notifications to other team members. |
| **14** | **Plan** In the planning phase, execute security analysis and create a plan to determine scenarios for how, where, and when testing will be done.  **Code** Deploy and use linting tools and Git controls to secure passwords and API Keys.  **Build** Use of Static application testing (SAST) tools to track down flaws in code before deploying it on production. These tools are specific to programming languages.  **Test** While testing your application, The dynamic application security testing (DAST) tools are used to detect errors associated with user authentication, authorization, SQL injection, and API-related endpoints.  **Release** The security analysis tools are used to perform vulnerability scanning and penetration testing. These tools should be used just before releasing the application.  **Deploy** After completing the above test in runtime, send a secure infra or build to production for final deployment. |