SOFTWARE DEVELOPMENT LIFE CYCLE

July 1st, 2024 version

Table des matières

Software Development Life Cycle Management

The TEEPTRAK Application Development Team adheres to a rigorous Software Development Life Cycle (SDLC) following to ensure the delivery of high-quality software solutions. The SDLC is structured into distinct stages, emphasizing verification and validation at each step.

1.1 Requirements Phase

The Requirements Phase involves the comprehensive collection and quantification of functional and non-functional requirements. This phase includes:

- **Stakeholder Engagement:** Conducting meetings and interviews with stakeholders to gather detailed requirements.
- **Requirements Documentation:** The Product Owner is responsible for creating detailed specifications that encompass business rules, functional processes, and security needs.
- **Scope Definition:** Clearly defining the project scope by identifying project deliverables, boundaries, and constraints. This includes documenting all functional and non-functional requirements to ensure alignment with business objectives and stakeholder expectations.
- **Change Management:** The Product Owner implements a structured process for managing changes to requirements. All proposed changes are documented, reviewed, and prioritised by the Product Owner in consultation with the CTO, COO, and CEO during weekly meetings. Approved changes are integrated into the project plan, and their impacts on scope, schedule, and resources are assessed and communicated to all stakeholders.
- **Security Requirements:** Including security requirements in the design phase to ensure that all aspects of security are considered from the start.

1.2 Analysis Phase.

During the Analysis Phase, the requirements gathered are systematically analysed to develop a logical model of the system. Key activities include:

- **Requirements Analysis:** Evaluating and refining the requirements to ensure clarity and feasibility.
- **Model Development:** Creating detailed report definitions, screen layouts, data element definitions, workflow diagrams, and security matrices.
- **Validation:** Ensuring that the logical model accurately reflects the requirements and business needs.
- **Security Checkpoints:** Establishing security checkpoints within the project milestones to ensure ongoing compliance with security standards.

1.3 Design Phase.

The Design Phase translates the logical model into a physical model, laying the groundwork for the actual development. This phase involves:

- **System Architecture:** Designing the overall system architecture, including business logic, database schemas, and object relationships.
- **Component Design:** Defining the design of individual components, including objects, report calculations, views, and security measures.
- **Detailed Specifications:** Producing comprehensive design documents that guide the subsequent coding phase.
- **Security in Design:** Incorporating security requirements into the design phase to ensure that the system architecture and components are secure from the ground up.

1.4 Implementation Phase.

The Implementation Phase is a critical stage where the system design is transformed into a functional software application. This phase encompasses the actual development, integration, and preparation for testing. Key activities in this phase include:

- **Development:**
    - Translating detailed design specifications into executable code.
    - Following best practices and coding standards to ensure code quality and maintainability.
    - **Secure Coding Guidelines:** Adhering to secure coding guidelines for each programming language used, including the use of free and open source libraries. Following the OWASP recommandations: https://cheatsheetseries.owasp.org/Glossary.html
- **Code Reviews and Automated Testing:**
    - Conducting peer reviews to ensure adherence to coding standards and to identify potential issues early. We have specific guidelines for each technical stack to ensure consistency and high quality across the project.
    - Utilising static code analysis tools to detect code smells and potential bugs.
    - Conducting unit tests to verify the functionality of individual components.
    - Performing integration tests to ensure that combined components work together correctly.
- **Security Compliance:**
    - Conducting security testing to validate the effectiveness of implemented security measures.
    - Utilising static code analysis tools to detect vulnerabilities, running each time that code is pushed on the software version control system, including dependencies.
    - Ensuring all communications with code repositories are encrypted using SSH.
- **Integration:**
    - Combining individual code modules and components developed by different team members.
    - Ensuring that the integrated application functions cohesively as a single system.
    - Addressing any integration issues and resolving dependencies.
- **Documentation:**
    - Maintaining comprehensive documentation of the codebase, including inline comments, function descriptions, and module overviews.
    - Updating software design documents to reflect any changes made during the implementation.
- **Pre-Deployment Preparation:**
    - Preparing deployment scripts and automation tools to streamline the deployment process.
    - Ensuring that the application is ready for the release phase, with all necessary configurations and dependencies in place.

This phase is crucial as it sets the foundation for subsequent testing and deployment activities. By adhering to rigorous development standards and thorough integration practices, the Implementation Phase ensures that the software is reliable, secure, and ready for comprehensive testing.

1.5 Testing Phase.

The Testing Phase ensures that the developed application meets the specified requirements and functions correctly. This phase encompasses:

- **Environment Setup:** Each developer conducts tests in their development environment before pushing code to the staging server.
- **Testing Stages:** Conducting component testing, requirements testing, and acceptance testing to identify and resolve defects.
- Quality Assurance:
  - **Staging Deployment:** Deploying the integrated application to a staging environment for comprehensive quality assurance.
  - **Functional Testing:** Conducting detailed functional tests to ensure that each feature works as intended.
  - **Regression Testing:** Performing regression tests to confirm that new changes have not adversely affected existing functionality.
  - **Performance Testing:** Assessing the application's performance to ensure it meets the required speed, scalability, and stability standards.
  - **User Acceptance Testing (UAT):** Engaging end-users to validate the application against business requirements and ensure it meets their needs.
  - **Defect Management:** Logging, tracking, and prioritizing defects identified during testing. Ensuring timely resolution and re-testing of fixed issues.
- **External Testing:** For major version updates, commissioning an external penetration test from an ANSSI/PASSI approved supplier to identify potential security breaches.
  - **Definition of a Major Version software update:** A release of a piece of software which is not merely a revision, a simple new feature or a bug fix release but which contains substantial changes (e.g., an overhaul of the main APIs, change in compatibility, change of back-end technology).

1.6 Production Phase.

The Production Phase involves deploying the tested application to the live environment. This phase includes:

- **Deployment:** Executing a general deployment plan to release new features or updates to users.
- **User Training:** Providing training and user guides to ensure effective use of the new features.
- **Server Updates:** Ensuring that on-premise servers are updated accordingly.

1.7 Maintenance Phase.

The Maintenance Phase focuses on the ongoing support and enhancement of the deployed application. This phase includes:

- **Scheduled Backups:** Regularly backing up the application to prevent data loss.
- **Issue Resolution:** Addressing any issues or bugs that arise post-deployment.
- **Continuous Improvement:** Implementing minor enhancements and planning for major updates.
- **Change Approval:** All changes are reviewed and approved by the CTO to ensure consistency and quality.
- **Daily Security Scans:** Performing daily security scans of the code and dependencies to identify vulnerabilities. Updating the code or upgrading dependencies accordingly to maintain security and compliance.

Definition of environments

2.1 Local Environment

The local environment consists of developers' computers, where initial development and testing take place. These machines are crucial for the productivity and security of the development process.

- Security Measures:
    - **Disk Encryption:** Ensuring all developer computers have encrypted disks to protect data in case of loss or theft.
    - **Password Protection:** Locking sessions with strong passwords to prevent unauthorized access.
    - **Secure Authentication:** Using multi-factor authentication to access development tools and repositories.
    - **Regular Updates:** Keeping operating systems and development tools up to date with the latest security patches.

2.2 Development Environment

The development environment is where individual developers the freshly written code. This environment is designed to be flexible and is configured to support the needs of the development team.

- Security Measures:
    - **Secure Authentication:** Ensuring only authorized developers have access to the development environment.
    - **Isolation:** Each developer's environment is isolated to prevent interference with others' work.
    - **Secure Code Repositories:** Using version control systems with secure access controls to manage code changes.
    - **Encrypted Communications:** Ensuring all communications with servers running the code are encrypted using HTTPS.

2.3 Staging Environment

The staging environment is used for integration testing and pre-production validation. It closely mirrors the production environment to ensure that testing results are accurate and reliable.

- Security Measures:
    - **Controlled Access:** Only authorized personnel can access the staging environment.
    - **Data Protection:** Using anonymized or synthetic data to protect sensitive information during testing.
    - **Environment Monitoring:** Continuously monitoring the environment for security vulnerabilities and breaches.
    - **Secure Hosting:** Staging servers and continuous integration instances are all stored on secure and isolated instances.
    - **Encrypted Communications:** Ensuring all communications with servers running the code are encrypted using HTTPS.

2.4 Production Environment

The production environment is the live environment where the application is deployed for end-users. This environment must be highly secure, reliable, and performant.

- Security Measures:
    - **Controlled Access:** Only authorised personnel can access the production environment.
    - **Environment Monitoring:** Continuously monitoring the environment for security vulnerabilities and breaches
    - **Regular Audits:** Conducting regular security audits and vulnerability assessments to identify and address potential security issues.
    - **Incident Response Plan:** Having a well-defined incident response plan to quickly address and mitigate security breaches.
    - **Secure Hosting:** Staging servers and continuous integration instances are all stored on secure and isolated instances.
    - **Encrypted Communications:** Ensuring all communications with servers running the code are encrypted using HTTPS.

Security awareness

- **Mandatory Security Training:** All employees must complete the SecNumAcademie training (https://secnumacademie.gouv.fr/) to ensure a baseline understanding of cybersecurity principles and practices.
- **Security Awareness Meetings:** The Chief Information Security Officer (CISO) conducts regular security awareness meetings to keep all employees informed about the latest security threats, best practices, and company policies. These meetings cover topics such as phishing prevention, secure password practices, data protection, and incident reporting procedures.