Table of Contents

[1.Introduction 3](#_Toc186297371)

[Project Description 3](#_Toc186297372)

[Project Structure 4](#_Toc186297373)

[2.SCM Plan Document 5](#_Toc186297374)

[2.1 Introduction 5](#_Toc186297375)

[2.1.1 Objectives 5](#_Toc186297376)

[2.1.2 Scope 5](#_Toc186297377)

[2.1.3 Key Terms 5](#_Toc186297378)

[2.1.4 References 5](#_Toc186297379)

[2.2 Management 5](#_Toc186297380)

[2.2.1 Organization 5](#_Toc186297381)

[2.2.2 Responsibilities 5](#_Toc186297382)

[2.2.3 Interface 5](#_Toc186297383)

[2.3 Activities 5](#_Toc186297384)

[2.3.1 Identification of Configuration Items 5](#_Toc186297385)

[2.3.2 Change Control Process 5](#_Toc186297386)

[2.3.3 Release Management 5](#_Toc186297387)

[2.3.4 Status Accounting 5](#_Toc186297388)

[2.4 Schedule 5](#_Toc186297389)

[2.5 Resources 5](#_Toc186297390)

[2.5.1 Tools and Techniques 5](#_Toc186297391)

[2.5.2 Personnel and Training 5](#_Toc186297392)

[2.5.3 Equipment 5](#_Toc186297393)

[Policies and Procedures 5](#_Toc186297394)

[3.SCM Process Simulation 11](#_Toc186297395)

# SCM Plan Document

## Introduction

### Objectives

* Ensure seamless configuration items (CIs) management for AH Artistry’s Makeup Service Booking System.
* Maintaining system stability and traceability.

### Scope

* Covers all CIs including source code, documentation, and deployment configurations related to AH Artistry’s Makeup Service Booking System.
* Focus on version control, change management, and maintaining audit trails.

### Key Terms

* Configuration Item (CI)
* Baseline
* Version Control
* Change Request
* Change Control Board (CCB)
* Configuration Control
* Configuration Status Accounting
* Configuration Audit
* Repository
* Build Management
* Release Management
* Traceability
* Environment
* Rollback
* Configuration Identification
* Scope
* Integration
* Documentation

### References

**Books**

* *Software Configuration Management Handbook* by Alexis Leon
* *Configuration Management Best Practices: Practical Methods that Work in the Real World* by Bob Aiello and Leslie Sachs
* *Software Engineering: A Practitioner's Approach* by Roger S. Pressman

**Standards**

* IEEE Standard 828-2012: IEEE Standard for Configuration Management in Systems and Software Engineering
* ISO/IEC/IEEE 12207: Systems and Software Engineering – Software Life Cycle Processes
* CMMI (Capability Maturity Model Integration)

**Articles and Papers**

* *An Approach to Software Configuration Management* (ACM Digital Library)
* *Effective Configuration Management for Software Development Projects* (Published in IEEE Software)

**Websites and Guides**

* SEI (Software Engineering Institute) - https://www.sei.cmu.edu/
* NASA Software Engineering Handbook - <https://software.nasa.gov/>
* PMBOK Guide by PMI (Project Management Institute)
* ITIL (Information Technology Infrastructure Library)

**Tools Documentation**

* Git Documentation - <https://git-scm.com/doc>
* Atlassian's Bitbucket and Jira Resources - <https://www.atlassian.com/>

## Management

### Organization

|  |  |
| --- | --- |
| **Roles** | **Name** |
| Configuration Manager | Ismairizz Bin Muhammad Rizal |
| Version Control Specialist | Shahril Aimar Bin Faizal |
| Change Control Manager | Aiman Harith Bin Abdul Halek |
| Audit Lead | Alwani Aqilah Binti Iskanda |
| Release Manager | Syamimi Binti Supian |
| Documentation Manager | Dayang Nur Alisa Binti Abang Senawi |

Table 2: Organization

### Table 2 above explains the roles and the people assigned to manage each specific responsibility in the Software Configuration Management (SCM) plan. Ismairizz Bin Muhammad Rizal oversees Configuration Items (CIs) as the Configuration Manager while Shahril Aimar Bin Faizal handles version control tasks like branching and merging. Aiman Harith Bin Abdul Halek manages change requests to ensure they are documented and approved. Other key roles include Alwani Aqilah Binti Iskanda conducting audits, Syamimi Binti Supian coordinating software releases, and Dayang Nur Alisa Binti Abang Senawi maintaining project documentation.

### Responsibilities

|  |  |
| --- | --- |
| **Roles** | **Responsibilies** |
| Configuration Manager | * Oversees the identification, organization, and control of Configuration Items (CIs). * Ensures adherence to SCM policies and processes across the team. |
| Version Control Specialist | * Manages version control tools (e.g., Git), including branching, merging, and tagging. * Maintains a stable repository and resolves conflicts effectively. |
| Change Control Manager | * Evaluates and approves Change Requests (CRs) in coordination with the Change Control Board (CCB). * Ensures that all changes are documented, tested, and implemented systematically. |
| Audit Lead | * Conducts regular configuration audits to ensure compliance with defined standards. * Verifies that all CIs are consistent, complete, and properly documented. |
| Release Manager | * Plans and coordinates the release of software versions to production. * Ensures that all CIs are tested and meet quality standards before deployment. * Manages deployment schedules and tracks post-release performance. |
| Documentation Manager | * Maintains and updates all project documentation, including SRS, user manuals, and test reports. * Ensures that all documentation is version-controlled and accessible to relevant team members. * Tracks changes to documentation alongside codebase updates. |

Table 3: Roles and Responsibilities

### 

### Interface

|  |  |  |
| --- | --- | --- |
| **From** | **To** | **Description** |
| Development Team | Configuration Manager | Updates on new code changes or configuration items to be added to version control. |
| Development Team | Quality Assurance Team | Sharing test results and defect reports for changes in configuration items. |
| Development Team | Operations Team | Collaboration on addressing deployment or environment-specific issues |
| Configuration Manager | Quality Assurance Team | Providing baselines and configurations for testing environments |
| Configuration Manager | Project Manager | Updates on the status of configuration items, baselines, and audits |
| Configuration Manager | Change Control Board | Submitting change requests for approval |
| Configuration Manager | Documentation Team | Sharing updated baselines and configurations for inclusion in user manuals or technical guides. |
| Configuration Manager | Audit and Compliance Team | Providing configuration status accounting records. |
| Configuration Manager | Tool Administrators | Coordination for maintaining and updating configuration management tools. |
| Project Manager | Stakeholders | Communicating changes in baselines, timelines, or scope due to configuration decisions. |
| Change Control Board | Stakeholders | Presenting impact analysis of changes. |
| Quality Assurance Team | Operation team | Ensuring proper handoff of tested builds for production deployment |

Table 4: Interface

## 

## Activities

### Identification of Configuration Items

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Configuration Item** | **Type** | **Location** | **Previous Version** | **New version** | **Description** |
| index.php | Source Code File | / | 1.0.0 | - | The main entry point of the website |
| config/ | Configuration Folder | /config/ | 1.0.0 | - | Folder containing configuration files |
| config.php | Configuration File | /config/connection.php | 1.0.0 | - | Defines database connection details |
| docs/ | Directory | /docs/ | 1.0.0 | - | Folder containing all documentation |
| SDS.docx | System Design Specification | /docs/SDS.docx | 1.0.0 | - | Documentation of AH\_Artistry System Design and Specification |
| SRS.docx | System Requirement Specification | /docs/SRS.docx | 1.0.0 | - | Documentaion of AH\_Artistry System Requirement Specification |
| src/ | Source Code Directory | /src/ |  | - | Folder containing source code files |
| server.php | Source Code File | /src/server.php | 1.0.0 | 1.0.1 | Backend logic for handling server-side operations.  v1.0.1: Bugfix: Resolve double booking issue |
| logout.php | Source Code File | /src/logout.php | 1.0.0 | - | Handles uer logout functionality |
| pages/ | Directory | /src/pages/ | 1.0.0 | - | Containing PHP Pages Source Code Files |
| about.php | Source Code File | /src/pages/about.php | 1.0.0 | - | Page for displaying about AH Artistry |
| appointments.php | Source Code File | /src/pages/appointments.php | 1.0.0 | - | - Page for displaying and handling appointments management.  - Accessible by admin (make-up artist) only.  - Main page for admin upon logged in. |
| book.php | Source Code File | /src/pages/book.php | 1.0.0 | - | Booking page which displays schedules and booking form |
| home.php | Source Code File | /src/pages/home.php | 1.0.0 | - | AH Artistry user homepage |
| login.php | Source Code File | /src/pages/login.php | 1.0.0 | - | AH Artistry login page for admin (make-up artist). |
| posts.php | Source Code File | /src/pages/posts.php | 1.0.0 | - | - Posts page which manages the posts-related actions functionality.  - Accessible by admin (make-up artist) only. |
| profile.php | Source Code File | /src/pages/profile.php | 1.0.0 | - | - Make-up artist profile page.  - Accessible by admin only. |
| css/ | Directory | /src/css/ | 1.0.0 | - | CSS folder containing website stylesheet files |
| style.css | Stylesheet File | /src/css/style.css | 1.0.0 | - | Defines the website styles |
| js/ | Directory | /src/js/ | 1.0.0 | - | Javascript folder containing javascript files (client-side functionality |
| main.js | JavaScript File | /src/js/main.js | 1.0.0 | - | Main Javascript logic for client-side functionality |
| phpmailer/ | Third-party Library | /src/phpmailer/ | 1.0.0 | - | PHPMailer library used for sending emails. No modifications to library files |
| img/ | Assets (Image Folder) | /img/ | 1.0.0 | - | Folder containing website images |
| posts/ | Image Folder | /img/posts | 1.0.0 | - | Folder containing images uploaded for posts |

Table 5: Identification of Configuration Items

### 2.3.2 Change Control Process

Policy: Changes must be reviewed, approved, and documented to avoid disruptions and maintain quality.

Procedure:

1. Submit a Change Request (CR) detailing:

* Description, rationale, impact analysis, and urgency.

1. Review the CR by the Change Control Board (CCB) comprising key stakeholders.
2. Approve, reject, or defer changes based on feasibility and alignment with project goals.
3. Implement changes following approval, updating relevant CIs and documentation.

### 2.3.3 Release Management

Policy: All releases must follow a structured process to ensure consistency, quality, and traceability. Releases should be approved by the Configuration Manager and the Change Control Board (CCB) before deployment.

Procedure:

1. **Planning**
   * Collaborate with all stakeholders to define the release scope and timeline.
   * Identify the Configuration Items (CIs) involved in the release, including new features, bug fixes, and documentation updates.
   * Prepare a deployment checklist to ensure completeness.
2. **Development and Testing**
   * Confirm that all changes have been implemented in the source code repository and adhere to version control policies.
   * Conduct rigorous testing in a staging environment to verify functionality, performance, and security.
   * Resolve any defects found during testing and retest as necessary.
3. **Approval**
   * Submit a release request to the Change Control Board (CCB) for evaluation.
   * Include test results, deployment plans, and a risk assessment in the submission.
   * Obtain approval from the Configuration Manager and CCB.
4. **Deployment**
   * Execute the deployment plan in the production environment following the deployment checklist.
   * Notify stakeholders, including the Development and Operations teams, about the release schedule and any expected downtime.
   * Monitor the release to ensure it performs as expected.
5. **Post-Release Evaluation**
   * Collect feedback from users and stakeholders to identify potential issues.
   * Monitor system performance and address any issues promptly.
   * Document the release process and lessons learned for future reference.
6. **Rollback Procedure** (if required)
   * Implement a rollback plan in case critical issues arise post-release.
   * Revert the system to the last stable state using backups and previous baselines.
   * Communicate with stakeholders about the rollback and address the identified issues.

Deliverables:

1. **Release Notes**:
   * Summarize the changes, including new features, resolved issues, and known limitations.
2. **Deployment Checklist**:
   * A detailed list of tasks completed before, during, and after deployment.
3. **Post-Release Report**:
   * A summary of the release process, feedback received, and issues encountered (if any).

### 2.3.4 Status Accounting

Policy: Maintain up-to-date records of all configuration items and changes to ensure project transparency.

Procedure:

1. Track the state of each CI such as active, under review, approved.
2. Log every change in a central database with details like date, author, and reason.
3. Generate periodic status reports summarizing recent changes, versions, and upcoming actions.
4. Share reports with stakeholders for review and decision-making.

## Schedule

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phase** | **Task** | **Start Date** | **End Date** | **Duration** |
| **Planning and Analysis** | Identify Client | 5 Nov 2024 | 6 Nov 2024 | 2 days |
| Gather Requirements | 7 Nov 2024 | 9 Nov 2024 | 3 days |
| Analyze Requirements and Feasibility | 10 Nov 2024 | 13 Nov 2024 | 4 days |
| Finalize Planning | 14 Nov 2024 | 15 Nov 2024 | 2 days |
| **System Architecture** | Define system Architecture | 16 Nov 2024 | 18 Nov 2024 | 3 days |
| Review Architecture with Team | 19 Nov 2024 | 20 Nov 2024 | 2 days |
| **Front-End Development** | Design UI/UX | 21 Nov 2024 | 25 Nov 2024 | 5 days |
| Develop Core Front-End Pages | 26 Nov 2024 | 3 Dec 2024 | 8 days |
| Implement Styling and Responsive Design | 4 Dec 2024 | 10 Dec 2024 | 7 days |
| **Back-End Development** | Set Up Database | 21 Nov 2024 | 25 Nov 2024 | 5 days |
| Develop APIs | 26 Nov 2024 | 7 Dec 2024 | 12 days |
| Integrate Back-End with Front-End | 8 Dec 2024 | 22 Dec 2024 | 15 days |
| **Testing and Debugging** | Perform Unit Testing | 23 Dec 2024 | 25 Dec 2024 | 3 days |
| Conduct Integration Testing | 26 Dec 2024 | 27 Dec 2024 | 2 days |
| Resolve Issues | 28 Dec 2024 | 28 Dec 2024 | 1 day |
| **User Acceptance Testing** | UAT Preparation | 29 Dec 2024 | 29 Dec 2024 | 1 day |
| Conduct UAT | 30 Dec 2024 | 30 Dec 2024 | 1 day |
| **Project Documentation** | Prepare Final Submission | 30 Dec 2024 | 30 Dec 2024 | 1 day |
| Submit Documentation | 31 Dec 2024 | 31 Dec 2024 | 1 day |

Table 6: Schedule

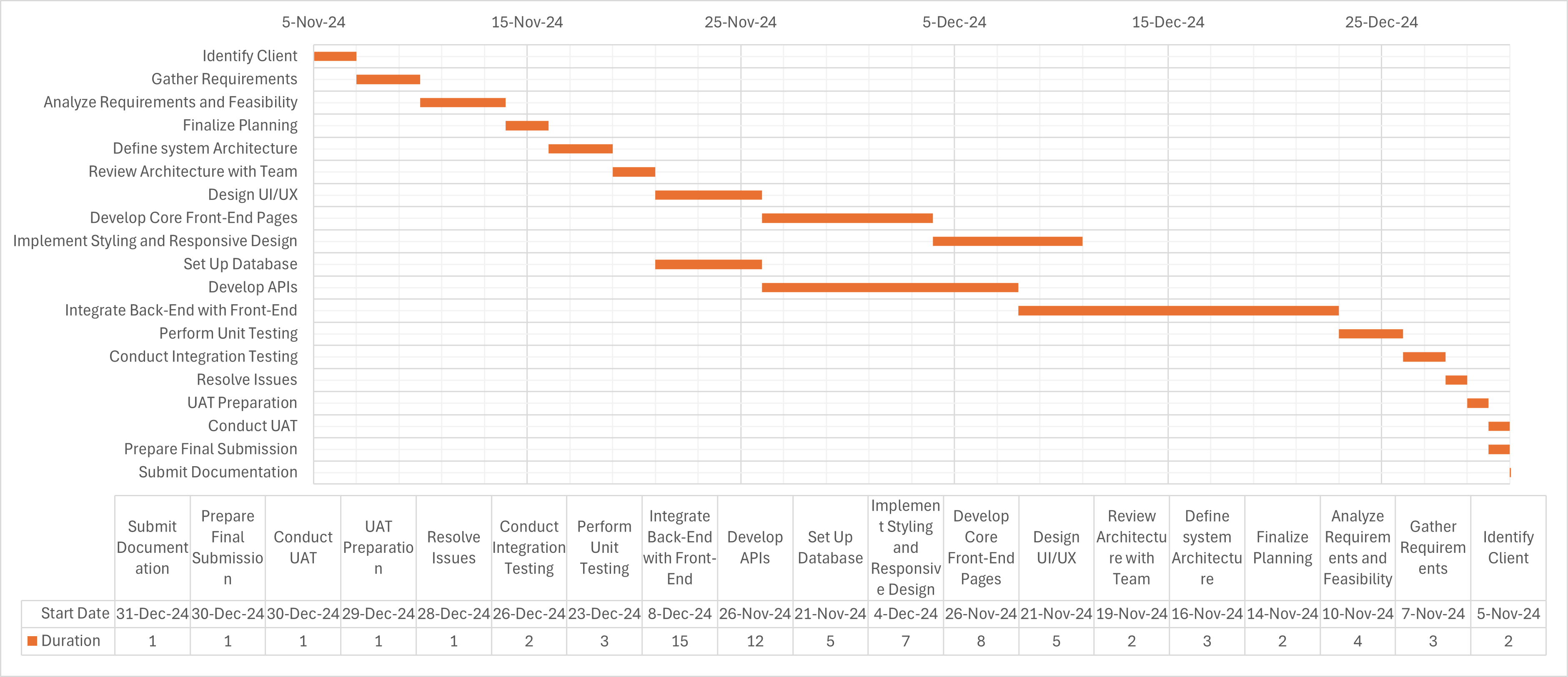


Figure 0.1: Gantt Chart.

## Resources

### 2.5.1 Tools and Techniques

**Configuration Management Tools:**

* **Atlassian's Jira**: Jira is utilized to manage configuration management processes, including tracking system changes, managing tasks, and facilitating team collaboration. It supports the maintenance of a Configuration Management Database (CMDB), which serves as a centralized repository for configuration items and their relationships. This ensures accurate tracking of system updates and enhances traceability throughout the project lifecycle.
* **Git**: Used for version control to track source code and documentation changes.

**Version Control System:**

* **Git**: A distributed version control system for tracking changes to source code and documentation.
* **Atlassian's Bitbucket**: A Git-based repository hosting service for version control and collaborative development.

### 2.5.2 Personnel and Training

**Configuration Manager**

**Qualifications:**

* Possesses qualifications in Computer Science, Information Technology, or a related field.
* Demonstrates strong knowledge of version control systems, build automation tools, and Configuration Management Database (CMDB) tools.
* Has experience with software development lifecycle (SDLC) processes and configuration management best practices.

**Responsibilities:**

* Oversee configuration management processes and tools to ensure efficiency and compliance with standards.
* Maintain and manage version control systems, including the creation and organization of release artifacts.
* Ensure accurate and detailed documentation to support traceability of changes.
* Collaborate with the development team, quality assurance, and stakeholders to facilitate smooth configuration and release management workflows.

**Release Manager**

**Qualifications:**

* Holds qualifications in Computer Science, Software Engineering, or a related field.
* Expertise in release planning, deployment strategies, and post-release monitoring.
* Experienced in coordinating cross-functional teams, risk management, and release scheduling.

**Responsibilities:**

* Plan and coordinate software releases, ensuring all Configuration Items (CIs) are tested and approved.
* Develop and execute deployment strategies to ensure minimal disruption to end-users.
* Monitor post-release performance and address any issues promptly.
* Communicate release plans and status updates to stakeholders and the development team.

**Development Team**

**Training Requirements:**

* Receive hands-on training in the use of Git (or similar version control systems) for effective version control and branching workflows.
* Gain familiarity with CI/CD tools, such as Jenkins, for building and deploying software efficiently.
* Develop an understanding of configuration management policies, best practices, and tools, including automation tools like Ansible.

### Equipment

* **GitHub: For version control and collaboration.**

**Purpose**: It helps teams manage source code changes, track the development history, and collaborate on code simultaneously.

**Key Features**:

1. **Branching and Merging**: Developers can work on individual branches, merge changes after review, and resolve conflicts efficiently.
2. **Conflict Management**: Ensures consistency by detecting and preventing conflicting changes in the code.
3. **Pull Requests**: A mechanism for reviewing and discussing code changes before merging them into the main branch.
4. **Collaboration**: Multiple contributors can work together seamlessly while maintaining code integrity.

* **JIRA: For tracking CRs and task assignments.**

**Purpose**: It is used to track change requests (CRs), assign tasks, and monitor project progress.

**Key Features**:

1. **CR Management**: Submit and track change requests with detailed descriptions, priorities, and status.
2. **Task Assignments**: Assign specific tasks to team members with deadlines and dependencies.
3. **Workflow Automation**: Custom workflows for approvals, reviews, and transitions.
4. **Integration**: Works with other tools like GitHub to link code commits directly to JIRA issues.

* **Draw.io: For creating workflow diagrams.**

**Purpose**: It helps design and document SCM workflows visually.

**Key Features**:

1. **Workflow Visualization**: Illustrates processes like version control, branching, and release management.
2. **Ease of Use**: Drag-and-drop interface for quickly building diagrams.
3. **Collaboration**: Enables real-time editing and sharing among team members.
4. **Integration**: Supports exporting diagrams to formats like PNG or embedding directly into reports.

* **VS Code: For source code management.**

**Purpose**: It is used for editing, debugging, and managing source code.

**Key Features**:

1. **Code Editing**: Write, format, and organize source code across various programming languages.
2. **Extensions**: Support for Git integration, making it easier to commit, pull, and push changes.
3. **Debugging Tools**: Built-in debugger to test and fix issues directly in the editor.
4. **Collaboration**: Extensions like Live Share allow for real-time collaborative coding sessions.

### Policies and Procedures

1. Configuration Items (CIs)

Policy: Every component critical to the software system such as code, documents, databases, and environment settings must be identified and tracked.

Procedure:

1. Assign a unique identifier and version to each CI.
2. Maintain a centralized repository with clear folder structures such as /src, /docs, /config.0
3. Use a defined naming convention such as “<name>.php\_v1.0.”
4. Version Control

Policy: All changes to CIs must be tracked using a reliable version control system to maintain consistency and traceability.

Procedure:

1. Use Git as the version control tool.
2. Implement branching strategies:

* main branch for stable releases.
* feature/<feature-name> branches for new features.
* bugfix/<description> branches for bug fixes.

1. Use Semantic Versioning:

* Major.Minor.Patch (e.g., 1.0.0 for initial release, 1.1.0 for new features, 1.1.1 for bug fixes).

1. Tag releases with version numbers for easy retrieval.
2. Configuration Audits

Policy: Regular audits must ensure all CIs comply with the defined standards and align with the baseline.

Procedure:

1. Conduct weekly audits to verify CI integrity and version accuracy.
2. Use checklists to assess:

* Correct versioning.
* Proper documentation of changes.
* Compliance with naming conventions and folder structures.

1. Report findings and address discrepancies immediately.
2. Process Documentation
3. Configuration Identification

Configuration Items (CIs):

* Code: HTML, CSS, JavaScript, PHP files.
* Documentation: SRS, user manuals, test reports.
* Data: Database schemas, and mock data for testing.
* Environment: Configuration files for local, testing, and production.

Naming Conventions:

* *Files*: Use lowercase with underscores such as booking\_system\_main.css.
* *Versions*: Semantic versioning such as v1.0.0.
* *Branches*: feature/ or bugfix/.

1. Repository Structure (Theoretical)

* */src*:

Contains source code organized by modules such as auth, booking, admin.

* */docs*:

Stores all documentation, including SRS and user manuals.

* */tests*:

Contains test scripts and reports.

* */config*:

Environment settings for development, testing, and production.