Software Lifecycle Management Guidelines

The purpose of these guidelines is to define and standardize the process for acquiring, implementing, and maintaining software in Government institutions in Rwanda.

- Introduction
- Scope and Objectives
- Overview
- Concept Note / Feasibility Study
    - Concept note for projects below USD 1 M [Mandatory]
    - Feasibility study for projects above USD 1M [Mandatory]
    - Approval process for concept note / feasibility study [Mandatory]
- Requirements Specifications and Terms of Reference
    - Software requirements [Mandatory]
    - Terms of reference [Mandatory]
- Software Acquisition
    - RISA framework contract [Mandatory]
    - Inhouse Development [Recommended]
    - External Procurement Process [Mandatory]
    - Request for Proposal [Mandatory]
    - Technical evaluation [Mandatory]

    - Financial evaluation[Mandatory]
    - Negotiations and Contracting [Mandatory]

- Architecture and Design
- Development
- Testing
    - Test planning [Mandatory]
    - Test design [Mandatory]
    - Test execution[Mandatory]
    - Test closure [Mandatory]

- Deployment
- Operations and Maintenance
- Upgrade or Decommission
    - Factors influencing the decision to upgrade or replace software
    - Measures to implement when upgrading software

- Software project management and delivery success factors
    - Planning and scoping [Recommended]
    - Vendor management [Recommended]
    - Governance, roles and responsibilities [Recommended]
    - Adopt agile software delivery approach [Recommended]
    - Adoption / change management plan [Mandatory]

Introduction

The purpose of these guidelines is to define and standardize the process for acquiring, implementing, and maintaining software in Government institutions in Rwanda. They provide leading practices for managing software at each stage of the software lifecycle with the objective of reducing software implementation risks leading to successful software projects.

In an increasingly digital era, government institutions are leveraging software systems to deliver services online, digitalize internal processes, improve efficiency, and foster transparency. The effective management of these software systems, from their initial conceptualization to their decommission is paramount to achieving these goals. This necessitates a robust approach to develop software lifecycle management guidelines ensuring that software systems are not only fit for purpose but also secure, sustainable, and deliver optimal value throughout their operational lifespan.

This document outlines guidelines to facilitate software lifecycle management specifically developed and implemented by government institutions, with emphasis on the procurement of software systems and inhouse developed systems from the point of project initiation, implementing and retirement.

These guidelines will empower government institutions to make informed decisions, foster innovation, and ultimately deliver superior digital services to the citizens they serve.

RISA serves as the central coordinating body mandated to provide oversight, standardization, and expertise to ensure that software development and acquisition aligns with national strategies including; (NST2 - the second generation of the National Strategy for Transformation (NST2) emphasizes the need to optimize governance structures to support efficient implementation, institutional framework, planning and coordination structures will be optimized and aligned with NST2 priorities thereby enabling faster and better service delivery.

Data protection law, ICT sector strategic plan, National data sharing policy, Blueprint digitalization transformation roadmap.), promotes interoperability, mitigate risks, and optimize public expenditure, by centralizing the governance of ICT project, we aim to prevent fragmented systems, reduce duplication of efforts, enhance cybersecurity.

RISA, in collaboration with government institutions, will take the lead in software systems procurement processes, and provide support throughout the implementation of software systems.

Scope and Objectives

> These guidelines cover the entire software lifecycle from initiation definition to decommissioning.

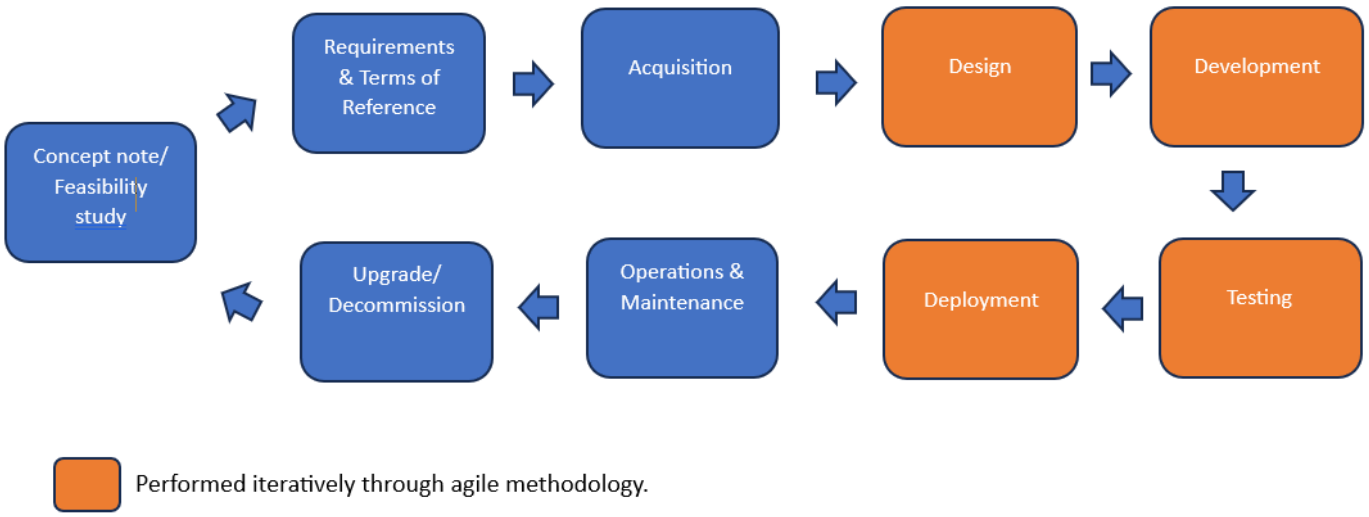
They are mandatory and applicable to all Government institutions in Rwanda and should be followed by Government staff as well as any external parties involved in the acquisition, implementation and maintenance of software within Government institutions. Key objectives of these guidelines include:

- Ensure software projects implemented in Government institutions are initiated and acquired in line with existing policies.
- Minimise duplication in acquisition and implementation of software projects across governments and ensure efficient sharing of resources wherever possible.
- Ensure software projects are implemented in line with best practices to ensure quality and ease of maintenance
- Ensure software projects meet the needs and requirements of institutions and deliver value
- Ensure software projects meet data protection and privacy regulations and requirements
- To secure the integrity and availability of government software and protect the institution from malicious attacks.
-

Overview

Software life cycle process [Mandatory]

Software Lifecycle Management refers to a structured process acquiring, implementing and maintaining high quality software. Government institutions should follow the following steps when deploying new software:



1. Concept note/feasibility study
2. Requirements Analysis and Terms of Reference
3. Software acquisition
   a. External Procurement
   b. Inhouse Development
4. Architecture & Design
5. Development
6. Testing
7. Deployment
8. Operations & Maintenance
9. Upgrading and Decommissioning

Agile approach [Recommended]

> The design, development, testing and deployment (Steps 4 to 7) should be done through an agile approach which is the recommended approach for software development in Government institutions.

Concept Note / Feasibility Study

In this phase the need for a new software project is identified through a situation assessment that defines the current problems or gaps, possible alternative solutions, the proposed solution option and goals or outcomes that need to be achieved. The identified project should be aligned with the sector specific strategies and with the institution's IT blueprint and roadmap. Depending on the expected value of the project, a concept note or feasibility study is required.

Concept note for projects below USD 1 M [Mandatory]

For projects expected to have a value below 1 million USD, a concept note is required.

The concept should have the following key components:

1. Situation assessment and identification of possible alternative solutions
2. Logical framework outlining the Project’s result chain through presentation of overall objective (impact), specific objectives (outcome(s)), outputs, and activity matrix with details on means (resource inputs) and costs
3. Justification of the selected technical option
4. Conceptual solution for the selected option
5. Detailed budget estimates for the project
6. Basic implementation plan and milestones
7. Identification of Funding and Cost Recovery Options
8. Project’s Socio-Economic Impact
9. Justification in relation to the National Planning Framework and sector objectives
10. Conceptual Study

Feasibility study for projects above USD 1M [Mandatory]

A feasibility study is required for projects whose expected value is above USD 1 million.

Key components of a feasibility study include:

1. Description of the context
2. Definition of objectives and logic of the intervention
3. Identification of the project
4. Demand and option analysis
5. Environmental and social considerations including Environmental and Social Impact Assessments
6. Technical design and cost estimates
7. Plan for implementation and - if applicable - operation of the project
8. Financial, economic and sensitivity analysis
9. Risk analysis
10. Conclusions and recommendations

Approval process for concept note / feasibility study [Mandatory]

The Chief Digital Officer/Head of IT Unit should take the lead in developing the concept note or feasibility study. The identified project should be in line with the institution's IT blueprint and roadmap.

The concept note or feasibility study should be approved by the Head of the institution/Chief Budget Manager before submission to RISA for approval along with the ICT Needs Planning Template.

For more details on the submission and approval process for new software projects please refer to the ICT Spend Control Guidelines for Public institutions.

> **Outputs:** Project concept note or Feasibility study

Requirements Specifications and Terms of Reference

This phase involves gathering the needs of the stakeholders and analyzing them to define detailed software requirements and Terms of Reference for acquisition of the software.

Software requirements [Mandatory]

Software requirements should include:

1. Functional requirements based on the needs of the institution’s operations and processes. This involves:

a. Reviewing “Existing” process flow
b. Developing “To be” process flows
c. Developing user stories based on the “To Be” process
d. Use case elicitations based on user stories

2. Technical requirements including:

a. Technologies supported or technology stack requirements
b. Minimum performance metrics
c. Reliability and availability requirements

3. Security, data privacy and risk requirements

Terms of reference [Mandatory]

Terms of reference define the project , objectives and the scope of work. They should include:

- Background information about the project and its objectives,
- Scope of work
- Software requirements/User Requirements Specifications
- Minimum requirements for the software vendor including experience and qualifications of resources needed to implement the project.
- Desired timelines for implementation

> **Outputs:** Terms of Reference for software acquisition, Software requirements specification document (SRS)

Software Acquisition

Reference is made to the ministerial instructions No 001/MINICT/20212 of 12/03/2012 related to the procurement of information and communications technology for goods and services by Rwanda public institutions providing guidance on leading procurement proceedings of ICT related supplies and other services on behalf of all government institutions by RDB which was later transferred to RISA.Click Here

RISA framework contract [Mandatory]

Once the project is approved as per the process defined in the ICT Spend Control Guidelines for Public institutions , the software can be developed inhouse or procured from an external vendor. For Government of Rwanda institutions software should be acquired through the RISA framework contract.

> In exceptional cases which require approval from RISA, external procurement and tender process may be followed.

Inhouse Development [Recommended]

An institution can internally develop and implement the solution if they have the required resources, skills and capacity. The agile approach is recommended for software development. Before opting for inhouse development an institution should consider the following:

- Assess the availability of skilled personnel and expertise within the institution for developing the software successfully.
- Consider long-term maintenance and support requirements and in-house capacity.
- Determine the scalability of in-house solutions to meet future needs.
- Analyze potential risks and challenges associated with in-house development.
- Explore interoperability with existing government systems and infrastructure.
- Evaluate the feasibility of collaboration with other agencies for shared solutions.
- Consider the time-to-market for in-house development versus using the RISA framework contract for outsourcing.
- Assess the impact on budget allocations and resource allocations within the agency.
- Evaluate security requirements to ensure compliance with data protection regulations.

External Procurement Process [Mandatory]

For externally procured software, RISA should be involved as it has the mandate of ensuring centralized software procurement. Government institutions should follow the following process for external procurement:

1. The institution submits the Terms of References (ToRs) including the Software Requirement Specifications to RISA.
2. RISA reviews the Terms of Reference and provides inputs and either approves or rejects the procurement request with reasons in consultation with the requesting institution.
3. If the Terms of Reference are approved, they are submitted to the designated service provider under the framework contract.
4. In the event the solution cannot be implemented under the framework contract, the institution shall officially request a no-objection from RISA to utilize other alternative options.
5. Once a no-objection is received from RISA, the institution will follow the RPPA public procurement process. The following should be considered at the various procurement stages.

Request for Proposal [Mandatory]

A request for proposal (RFP) document is developed based on the Terms of Reference and software specifications. Evaluation criteria should also be developed to guide the evaluation process. The RFP and evaluation criteria should cover the following:

1.Technical capabilities of the vendor. This includes:

a. Experience of the vendor in successfully delivering similar projects
b. Experience and qualifications of the proposed vendor team

2. Strengths of the proposed solution. This includes:

a. Fit of the proposed solution and how it addresses the institution’s requirements
b. Proposed methodology and approach for implementation

3. Financial proposal which includes assessing the total cost of ownership of the proposed software

> **Outputs:** Request for Proposal, Evaluation criteria

Technical evaluation [Mandatory]

Technical evaluation involves evaluating the technical proposal based on the evaluation criteria. Where possible this can include requesting vendors to demonstrate their solutions as well as making reference calls or visits to validate vendor’s experience.

Financial evaluation[Mandatory]

Financial evaluation[Mandatory]

Financial proposals should be evaluated as per the evaluation criteria and in accordance with the Public Procurement guidelines. When evaluating the cost of the proposed solutions, it is important to consider total cost of ownership including acquisition or development cost, any applicable licenses, maintenance and support costs.

> **Outputs:** Tender Evaluation Report

Negotiations and Contracting [Mandatory]

Contracting process involves the following:

1. **Negotiations -** Negotiations with the selected vendor covering the scope of work and financial proposal.
2. **Contract drafting:** The contract should be drafted by the institution's legal team and technical input from IT team. The contract should include the agreed terms and conditions of the software procurement, including the price, the scope of work, the support and maintenance terms and the termination terms.
3. **Contract review:** The contract should be reviewed with the vendor to ensure that the terms and conditions are acceptable to both parties. Contract review process should follow the Public Procurement guidelines.
4. **Contract signing:** The contract should be signed by both parties before the software project commences.

> **Outputs:** Vendor Contract

The section below details the stages of the software development process. While these are shown sequentially, they can be implemented using the agile methodology.

Architecture and Design

The Architecture and Design phase involves transforming the software specifications into a technical design. This phase involves designing the software architecture, user interface, the database and integration interfaces. The activities to be carried out in this phase include:

- **Requirements validation[Mandatory]** - Software requirements and specifications should be validated to ensure that they are clear and understood prior to design.
- **Software architecture design [Mandatory]** - Defining the software architecture by identifying the components of the system, their relationships, and how they interact with each other.
- **UI/UX Design [Mandatory]**- Design the user interface by defining the look and feel of the system, as well as the way that users input data and interact with the system. Refer to RISA UI/UX and accessibility guidelines
- **Database design [Mandatory]** - Designing the database by defining the structure of the data, as well as the way that data is stored and retrieved.
- **Integration design [Mandatory]** - Design t any required integration interfaces based on the approach or standards agreed
- **Design review [Mandatory]** - Perform a design review, which includes technical reviews of application and infrastructure, as well as a review of high-level processes.
- **Design of training [Mandatory]**- Design initial end-user training and awareness programs.
- **Infrastructure Design [Mandatory]** - Design the infrastructure required including establish separate development, test and production environments.

> **Output:** System Architecture Document, User Interface (UI) Design, Integration Design, Database Design

Development

The Development phase involves the process of transforming the design into a working software system. This phase involves coding the software, unit testing it, and integrating it with other components of the system. Institutions should refer to RISA guidelines for web and mobile applications development. Key guidelines include:

- **Clear understanding of requirements and design[Mandatory]** - Developers should ensure clear understanding of the requirements and design with frequent engagement with stakeholders including Project managers, business analysts and users
- **Version Control [Mandatory]** - Utilize version control systems such as Git to manage source code efficiently. This enables collaboration among developers, facilitates tracking changes, and provides a backup of the codebase.
- **Automated Testing [Recommended] -** Implement automated testing where possible for unit, integration, and regression testing. This ensures code quality, detects bugs early, and facilitates continuous integration and delivery.
- **Code Reviews[Recommended] -** Conduct regular code reviews to ensure code quality, adherence to coding standards, and knowledge sharing among team members. Code reviews also help identify potential issues and promote better design practices.
- **Modular Design[Recommended] -** Follow modular design principles to break down the software into smaller, manageable components. This promotes code reusability, maintainability, and scalability.
- **Documentation[Mandatory] -** Maintain comprehensive documentation for the software, including code comments and API documentation. Good documentation improves understanding, facilitates collaboration, and eases maintenance.
- **Continuous Integration/Continuous Deployment (CI/CD)[Recommended] -** Implement CI/CD pipelines to automate the build, testing, and deployment processes. This ensures rapid and reliable delivery of new features and updates while maintaining stability.See DevOps guidelines
- **Security Practices[Mandatory]** - Integrate security measures throughout the development process, including secure coding practices, vulnerability scanning, and penetration testing. Security should be a priority from the early stages of development to mitigate risks. See RISA security by design guidelines
- **Team Collaboration and Communication[Recommended]** - Foster a collaborative and communicative environment within the development team. Regular meetings, stand-ups, and communication channels facilitate coordination and alignment towards common goals.

> **Output:** Source code, Software deployed in Test environment

Testing

Testing is the process of verifying and validating that the software meets the requirements of the institution and its stakeholders. This phase involves a variety of testing activities, such as unit testing, system testing, integration testing, user acceptance testing, performance testing and security testing.

The activities to be carried out in this phase are detailed below.

Test planning [Mandatory]

Test planning [Mandatory]

The objective of test planning is to define the scope, approach, resources and schedule of testing activities. It includes:

1. Defining test scope and objectives
2. Defining the required resources including testing team, infrastructure, logistics and testing tools
3. Defining the approach for testing including the entry and exit criteria for each testing stage
4. Defining the expected test deliverables
5. Scheduling test activities in an appropriate manner

> **Output:** Test strategy and test plan

Test design [Mandatory]

This step involves defining the test scenarios required to test the software requirements and the test cases required to test each scenario. Test cases should include:

1. The reference to the requirement being tested
2. Description of the function or feature being tested
3. Test data & instructions to run the test
4. Expected test results

> It is crucial to keep in mind that tests should be designed to test both the specified and unwritten requirements including expected user behavior and arbitrary scenarios that could break the system.

> **Output:** Test scenarios and test cases

Test execution[Mandatory]

In this step testing is executed based on the test plan and test cases. Ensure the testing teams are aware of their responsibilities prior to testing. Ideally the testing team should be in one location to enable easier collaboration. Key activities include:

1. Preparation and setup of the test environment
2. Execution of test cases based on the test plan
3. Recording of test results
4. Recording, classification and prioritization of identified defects
5. Resolution of defects based on priority
6. Regression testing after defects have been resolved

The minimum test types to conduct include

- Unit testing
- Integration end-to-end (e2e) tests
- User acceptance tests
- Security testing as per software security guidelines
- Performance / Stress tests

> **Output:** Test reports with test completion status and test pass rates, Defect logs

Test closure [Mandatory]

This is the final step in the testing process. The testing phase is closed when the following is achieved:

1. All the tests have been completed and there are justifications for any tests that cannot be completed
2. All critical defects have been resolved. If any defects are open, then reasons for defects being open should be provided. Defects should be analysed based on their impact on the operations of the institution. A decision should be made on which defects must be closed before the software or a component of the software can go to production

> **Output:** Final testing report showing completion rate, pass rate and status of defects

Deployment

Deployment is the process of transitioning the new software into production and making it available to users. This phase involves activities such as installing the software, configuring it, and training users on how to use it.

The activities to be carried out in this phase include:

- **Pilot deployment[Recommended]** - Conduct a pilot/test deployment of the infrastructure, application and other relevant components. Conduct transition between pilot and full-scale deployment.
- **Software Configuration[Mandatory]** - Configure the software which involves setting up the software to meet the specific needs of the users, such as the data that will be stored, the security settings and user permissions.
- **Secure data migration [Mandatory]** -Implement appropriate safeguards to protect personal data during transit and storage, such as encryption and secure communication protocols
- **Training [Mandatory] -** Deploy training and awareness programs to train end users and technical personnel who will support and maintain the software
- **User support [Mandatory]** - Provide a way of delivering user support where users can report issues, request changes or ask for help while using the software.

> **Outputs:** Technical system documentation, User documentation

Operations and Maintenance

The Operations and Maintenance (O&M) phase involves keeping the software up and running after it has been deployed. This phase involves activities such as monitoring the software, fixing bugs, and making changes to the software as needed. The activities to be carried out in this phase include:

- **Monitor the software[Recommended]** - Monitor performance of the software to ensure it continues to operate optimally. Use available tools to monitor availability and performance
- **Ongoing user training and support[Mandatory] -** Ensure there is a process for conducting ongoing training and awareness as well providing channels where users can report any issues faced with the software. Channels can include an online help desk system and telephone contact. Issues should be logged, classified and resolved in a prioritized manner
- **Change management[Mandatory]** - Changes to software should follow a systematic process with change requests documented and approved before changes are made.
- **Maintaining Software Security[Mandatory]** - Implement security measures such as firewalls and intrusion detection systems, periodic system vulnerability assessments and installing security updates and patches. Refer to RISA security by design guidelines.
- **Backup and disaster recovery[Mandatory**] - Ensure there is a disaster recovery process to ensure the institution's data is not lost should an incident occur.

> **Outputs:** Software support issue logs, Updated user and technical documentation

Upgrade or Decommission

The Upgrade or Decommissioning phase involves retiring or removing a software system from service. The software may then be replaced by new or upgraded software. This phase involves activities such as archiving the software, disposing or repurposing of the hardware, and notifying the users.

Factors influencing the decision to upgrade or replace software

Software may need to be upgraded when it starts impacting efficiency, productivity, scalability, security, user experience or compatibility with modern technology. Some of the factors that may result in the need to upgrade software include:

- **No longer meeting institution’s requirements:** Software may need to be upgraded when its capabilities no longer meet the evolving needs and requirements of the organization and it may be more costly to enhance the existing system or it may not be technically feasible.
- **Technology trends:** If the software does not support current industry trends it may need to be upgraded. Examples can include incompatibility or inability to integrate with newer hardware or software systems. In addition newer versions of software may have features and capabilities not supported by the existing software.
- **Challenges with performance and scalability:** If the system has persistent challenges with performance that impact the experience of its users. These are performance issues with the software that are not related to underlying infrastructure and that cannot be easily resolved. This can also be the case if the software is not scalable to the growing needs of the institution.
- **End of support:** Software provided by external vendors may reach its end of life or end of support. This means the software is no longer supported by the vendor and critical updates such as software enhancements and security patches are no longer available. This would pose a risk to the institution and institutions need to monitor end of life dates and plan for upgrades in advance.
- **Cost/ benefit analysis:** when the cost of maintaining the system exceed the benefits and upgrading provides a better cost/benefit outcome in the longer term.

Measures to implement when upgrading software

The following should be considered when upgrading software to minimize the risks involved in the process such as data loss or security risk.

- **Impact analysis [Mandatory] -** Conduct an analysis to fully understand the impact of decommissioning the software. This includes an understanding of impact to operations of the institution and users as well as cataloging all internal and external systems and interfaces that integrate with the current system. Risks and mitigations should also be identified and document.
- **Secure the data[Mandatory]** - Data should be secured from loss or unauthorized exposure. The institution should determine what data needs to be migrated to new systems or archived based on the institution's data retention requirements as well any legal requirements. This should be documented as part of the data migration strategy. Data should be securely backed up prior to decommissioning of the software and any data being migrated should be protected from unauthorized access. Hardware should also be properly cleansed of the old software and any sensitive data.
- **Software archival [Mandatory] -** Consider if the software needs to be archived for future reference. If required, the software being replaced should be secured by copying the software to a secure location so that it can be accessed if needed in the future. This should be done in accordance with any license or IP requirements.
- **Stakeholder communication [Mandatory] -** All relevant stakeholders such as user departments and IT teams should be involved in the decision to replace the software and should be informed of the transition plans. They should also be informed of how their data will be managed and retained during the transition process.
- **Documentation update [Mandatory]-** Any documents in the organization referring to the replaced software such as policies and procedures should be updated accordingly.

> **Outputs:** Business case/justification for upgrading or replacing the software, Impact analysis including risks and mitigations, Data migration strategy covering data security controls, data migration and archival plans, Stakeholder communication plan, Archived software and data

Software project management and delivery success factors

Software projects can be complex and with a high failure rate and need effective management to increase the chance of success. Key success factors in delivering software projects include:

Planning and scoping [Recommended]

- The scope of the project should be realistic and clearly defined
- Scope management procedures should be put in place to ensure that requests for scope changes are analysed for impact on budget, timelines and resources requirements before they are approved for implementation
- The project plan should be realistic and factor in the scope of the project, resources available and external dependencies

Vendor management [Recommended]

- Evaluation criteria for vendors during the procurement process should factor in their prior experience and qualifications and strength of their team. In addition to evaluation of desktop proposals, wherever possible vendors should demonstrate their existing solutions/previous work and reference calls or visits done to validate prior experience.
- Vendor scope should be clearly defined in the contract
- Project planning should be a joint activity with the vendor to ensure plans are realistic and adequately resourced
- Payment milestones in the vendor contract should be aligned with verifiable progress of deliverables
- Have regular progress reviews on progress against the milestones with timely management of any risks and issues identified as per the governance process adopted

Governance, roles and responsibilities [Recommended]

- The governance and decision-making process for the project should be clearly defined including roles for the project sponsor, steering committee, project manager, internal project team and external vendors if applicable
- Ensure adequacy and availability of the internal project team based on their expected roles. Consider outsourcing where there are gaps that cannot be filled internally

Adopt agile software delivery approach [Recommended]

Agile software development is the recommended approach for Government institutions. The iterative nature of the approach helps to manage complexity, ensure timely feedback from stakeholders, achieve incremental delivery of software where feasible and enable continuous improvement in the delivery process.

Adoption / change management plan [Mandatory]

New software projects often lead to new ways of working for users and key stakeholders. Change management is therefore important to ensure a high level of engagement of staff and other project stakeholders leading to greater adoption and success of new software projects. Key considerations for change management include:

- Involving users early on the project starting with the project concept phase. Users should be involved in defining needs and gaps that need to be addressed by the project and this should not be left to just the IT teams
- A change impact assessment should be carried out during the design phase of the project to assess what impact the project has to the key stakeholders and what they need to be able to adjust to the plan
- A communication plan should be developed based on the needs of each stakeholder to ensure they are regularly informed about the progress and what is required of them, and that there is a mechanism to get their feedback as the project progresses
- Involve key users in the testing and acceptance process for the project based on their respective roles and responsibilities
- A training plan should be developed for the different users to ensure they able to adjust smoothly to new software and new ways of working
- Ensure there is an effective support process post implementation to support users should they run into challenges when transitioning to the new project