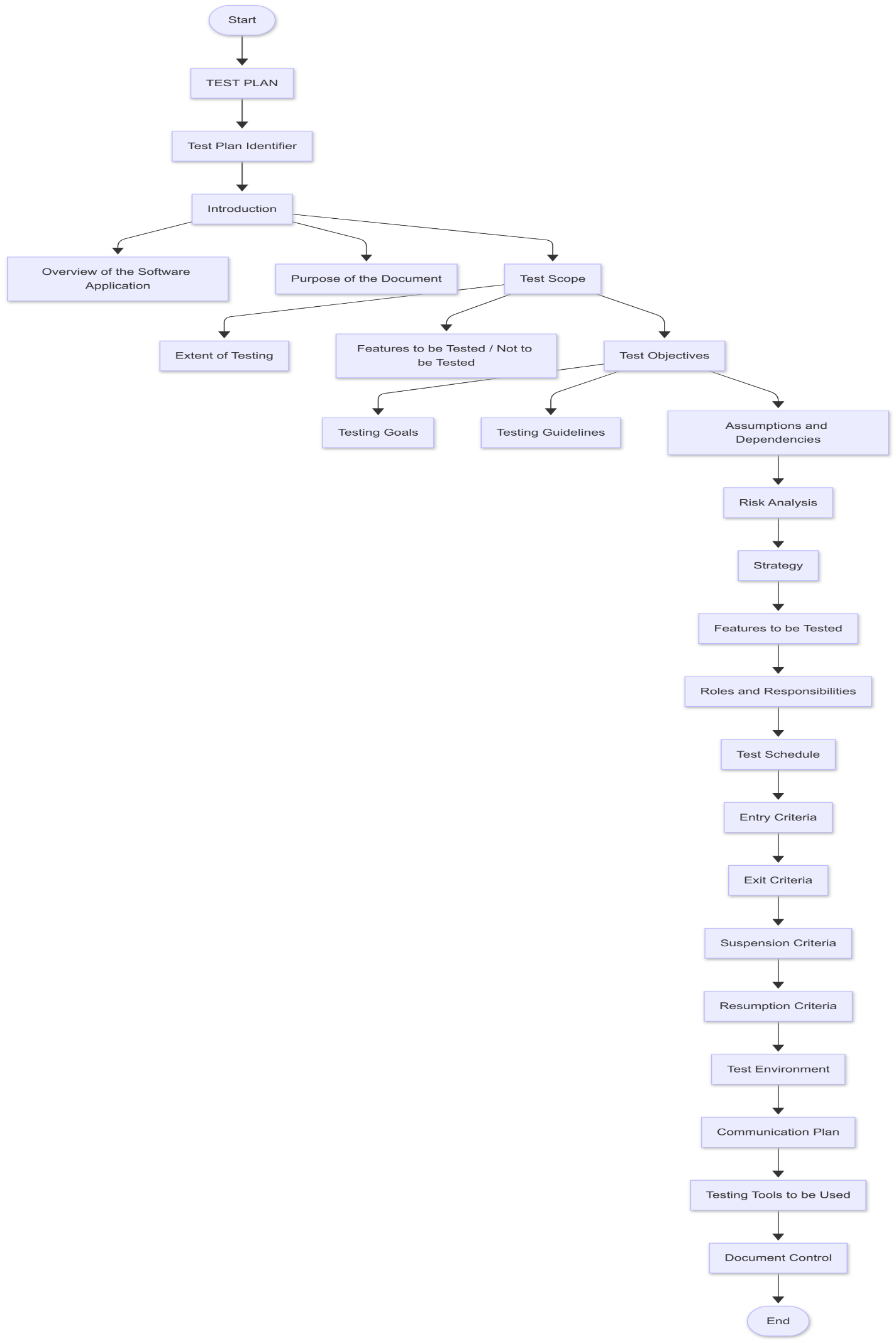
**TEST PLAN**



**1. Test Plan Identifier**

* This is the name or ID of the test plan.
* It helps to track different versions of the plan.
* It can include project name, version, and date.
* Useful for referencing in meetings or documents.
* Example: “TP\_MobileApp\_v1.0\_2025”.

**2. Introduction**

* Gives a brief idea of what the document is about.
* Describes the software project being tested.
* Tells who the audience of the test plan is.
* Sets the context for the rest of the document.
* Helps new team members quickly understand the project.

**3. Overview of the Software Application**

* Summarizes what the software is and what it does.
* Includes major features and purpose of the app.
* May mention users (like customers, admins).
* Tells the platform — like web, mobile, or both.
* Gives background for understanding the testing focus.

**4. Purpose of the Document**

* Explains why this test plan was created.
* Describes how this document helps the QA process.
* Tells the reader what to expect from the test plan.
* Ensures everyone is aligned on the testing approach.
* Helps in planning, reviewing, and executing tests.

**5. Test Scope**

* Defines what features and functions will be tested.
* Tells what is not in scope (what we will skip).
* Helps avoid confusion about coverage.
* Useful for time and resource planning.
* Helps stakeholders understand the testing boundaries.

**6. Extent of Testing**

* Describes how deep and wide the testing will go.
* Says if we are doing full or partial testing.
* Mentions if it's only UI testing, or includes backend too.
* Shows whether performance or security will be tested.
* Helps QA decide test case volume and effort.

**7. Features to be Tested / Not to be Tested**

* Lists which modules or features we will test.
* Also mentions any features that we will not test now.
* Reasons for not testing could be time or dependencies.
* Helps avoid testing areas that are not ready.
* Brings clarity to QA, Dev, and Management teams.

**8. Test Objectives**

* States what we want to achieve with this testing.
* The main goal is to find bugs and ensure quality.
* Confirms the product works as per requirements.
* Ensures all main user journeys are verified.
* Helps the team focus on expected results.

**9. Define a “Testing Goal”**

* Example: "Ensure login and payment features work properly".
* Clear goals guide test case design.
* Goals keep the team aligned and focused.
* They help measure testing success.
* They’re usually linked to key features of the app.

**10. Provide the Guidelines of Testing**

* Explain how tests will be written (e.g., using test case template).
* Mention how bugs will be reported (e.g., in JIRA).
* Specify rules for naming, tagging, and documenting.
* Follow best practices for test case design.
* Helps maintain consistency across the team.

**11. Assumptions and Dependencies**

* Things we believe to be true for testing to work.
* Example: "Test data will be available on time".
* Dependencies are things outside our control (like APIs).
* Knowing these helps plan testing better.
* If assumptions fail, testing can be delayed or affected.

**12. Risk Analysis**

* Identifies possible problems before or during testing.
* Examples: build delays, unclear requirements.
* Helps us prepare backup plans.
* Risks are ranked by impact and likelihood.
* Important for project managers to control timelines.

**13. Strategy**

* Explains the testing approach (manual, automation, both).
* Says which types of testing will be done — like UI, API, regression.
* Lists test levels — like unit, system, UAT.
* Helps in tool selection and resource planning.
* Provides a roadmap of how testing will happen.

**14. Features to be Tested**

* Lists all the parts of the app we’ll test.
* Each listed feature should have test cases.
* Features usually match user stories or modules.
* Ensures nothing important is missed.
* Tells devs what areas will be reviewed for quality.

**15. Roles and Responsibilities**

* Lists all team members involved in testing.
* Assigns roles — like test case writer, reviewer, automation lead.
* Helps avoid confusion in tasks.
* Ensures accountability.
* Makes collaboration smooth and clear.

**16. Test Schedule**

* Timeline for testing activities.
* Shows when each phase (like test case writing, execution) starts and ends.
* Helps in tracking progress.
* Keeps QA in sync with development timelines.
* A delay here affects the whole release plan.

**17. Entry Criteria**

* What must be ready before testing begins.
* Examples: Stable build, environment setup, test data ready.
* Prevents wasting time on broken or incomplete builds.
* Ensures QA starts only when ready.
* Reduces early test failures.

**18. Exit Criteria**

* When testing is considered complete.
* Examples: All test cases passed, no critical bugs.
* Helps decide release readiness.
* Prevents rushing incomplete testing.
* Should be agreed on by QA and project leads.

**19. Suspension Criteria**

* Conditions to temporarily stop testing.
* Example: Build is crashing or feature not working.
* Prevents wasting effort during major issues.
* Helps focus on fixing first.
* Once fixed, testing can resume.

**20. Resumption Criteria**

* Conditions to restart testing after a pause.
* Example: Bug fix confirmed and build updated.
* Ensures issue is resolved before testing restarts.
* Testing resumes smoothly from where it stopped.
* Keeps testing quality consistent.

**21. Test Environment**

* Where and how testing will be done.
* Example: Windows 11, Chrome browser, test server.
* Should match the real user setup as much as possible.
* Includes tools, network, and backend systems.
* A stable environment helps find real bugs.

**22. Communication Plan**

* How the team will communicate (daily calls, JIRA, email).
* Who reports progress and to whom.
* Defines reporting formats (like test summary).
* Keeps everyone informed about testing status.
* Helps resolve issues quickly.

**23. Testing Tools to be Used**

* Lists all tools used during testing.
* Example: Postman for APIs, Selenium for automation.
* Each tool must serve a clear purpose.
* Tools improve speed and accuracy.
* Team should be trained on tools before use.

**24. Document Control**

* Tracks version history of the test plan.
* Shows who wrote and who approved the document.
* Adds last updated date.
* Ensures the team uses the latest version.
* Important for audits or future reviews.