**Basic Study Material on Software Test Management Using Zephyr Scale and Jira**

**1. Introduction to Software Test Management**

Software test management is an essential process in software development that involves planning, tracking, and managing test cases to ensure software quality. A good test management tool helps teams organize their test cases efficiently, track progress, and generate reports.

**2. Why Not Use Google Sheets for Test Case Management?**

Some teams use Google Sheets to write and track test cases, but this approach has limitations:

| **Feature** | **Google Sheets** | **Test Management Tools (e.g., Zephyr Scale)** |
| --- | --- | --- |
| Traceability | No | Yes |
| Test Cycle Management | No | Yes |
| Automated Reporting | No | Yes |
| Integration with Jira | No | Yes |

Due to these limitations, professional teams prefer dedicated test management tools like **Zephyr Scale**, which integrates seamlessly with **Jira**.

**3. Introduction to Zephyr Scale**

**Zephyr Scale** is a widely used test management tool that allows software testers to create, manage, and execute test cases effectively. It provides:

* **Traceability:** Links test cases to requirements.
* **Test Cycle Management:** Organizes test cases into structured test cycles.
* **Reporting:** Generates reports on test execution and results.
* **Automation Support:** Integrates with automation tools.

**4. Setting Up Zephyr Scale with Jira**

**Step 1: Sign Up for Jira**

1. Visit [Atlassian Jira](https://www.atlassian.com/software/jira)
2. Click on **Get It Free**
3. Use a **valid email** (either personal or work email)
4. Follow the email verification process

**Step 2: Creating a Jira Project**

1. Log in to Jira.
2. Click on **Create Project**.
3. Select **Scrum Project** (Scrum is a framework for Agile development).
4. Name your project (e.g., **Project 1**).
5. Click **Create**.

**Step 3: Adding Zephyr Scale to Jira**

1. Click on **Apps** in Jira.
2. Select **Explore More Apps**.
3. Search for **Zephyr Scale**.
4. Ensure you select **Zephyr Scale** (not Zephyr Squad).
5. Click **Try It Free** (Free for teams of up to 10 users).
6. Wait for installation to complete.
7. Navigate to your project and verify that **Zephyr Scale** is installed.

**5. Understanding Test Cases in Zephyr Scale**

A **test case** is a set of steps to verify a specific functionality in a software application.

**Example of a Test Case**

| **Test Case ID** | **Description** | **Steps** | **Expected Result** |
| --- | --- | --- | --- |
| TC\_001 | Login to the system | 1. Open login page 2. Enter username 3. Enter password 4. Click login | User is logged in successfully |

Using Zephyr Scale, testers can:

* Create test cases.
* Organize them into **test cycles**.
* Execute tests and log results.
* Generate reports for stakeholders.

**6. Managing Test Cycles in Zephyr Scale**

A **test cycle** is a group of test cases executed together during a testing phase.

**Steps to Create a Test Cycle in Zephyr Scale**

1. Open Jira and navigate to your project.
2. Click **Zephyr Scale** > **Test Cycles**.
3. Click **Create Test Cycle**.
4. Add test cases to the cycle.
5. Assign testers.
6. Execute test cases and log results.

**7. Reporting and Traceability in Zephyr Scale**

Zephyr Scale provides built-in reporting features, allowing teams to:

* Track test execution status.
* View test coverage.
* Generate detailed reports.

**Example Report**

| **Test Cycle** | **Passed** | **Failed** | **Blocked** |
| --- | --- | --- | --- |
| Cycle 1 | 10 | 2 | 1 |
| Cycle 2 | 15 | 3 | 0 |

**8. Common Issues and Solutions in Zephyr Scale**

| **Issue** | **Solution** |
| --- | --- |
| Zephyr Scale is not loading | Refresh the page or wait for installation completion |
| Cannot find Zephyr Scale in Jira | Ensure you installed **Zephyr Scale**, not Zephyr Squad |
| Test cases not saving | Check for network issues and reattempt |

**9. Summary**

* **Zephyr Scale** is a powerful test management tool integrated with **Jira**.
* It allows teams to **write, organize, execute, and report** test cases effectively.
* Using **test cycles**, teams can streamline their testing workflow.
* Proper integration and configuration ensure smooth usage.

This study material provides a basic understanding of software test management using Zephyr Scale and Jira, suitable for beginners looking to enter the field of software testing.

**Study Material: Introduction to Zephyr Scale and Test Case Management in Jira**

**1. Introduction to Zephyr Scale**

Zephyr Scale is a test management tool integrated into Jira that helps teams plan, execute, and track software testing. It provides a structured approach to managing test cases, test cycles, and test plans.

**Key Features of Zephyr Scale:**

* Organize test cases in folders
* Execute test cases and track results
* Link test cases with user stories for traceability
* Generate detailed reports

**2. Exploring Zephyr Scale in Jira**

When you access Zephyr Scale within a Jira project, you will find the following key components:

1. **Test Cases** – Individual test scenarios to validate system functionality.
2. **Test Cycles** – Groups of test cases that are executed together.
3. **Test Plans** – A collection of test cycles planned for execution.
4. **Reports** – Summarized test execution results.

**Diagram: Zephyr Scale Structure**

+----------------+

| Test Plans |

+----------------+

|

|

+----------------+

| Test Cycles |

+----------------+

|

|

+----------------+

| Test Cases |

+----------------+

**3. Creating a Test Case**

A test case is a set of conditions used to determine if a feature in the application is working correctly.

**Steps to Create a Test Case:**

1. **Go to Zephyr Scale** → Select "Test Cases"
2. Click "Create Test Case"
3. Fill in the following details:
   * **Name:** e.g., "Verify login with valid credentials"
   * **Objective (Optional):** Purpose of the test
   * **Precondition (Optional):** Requirements before testing (e.g., "User must be registered")
   * **Status:** Draft, Approved, or Deprecated
   * **Priority:** High, Medium, or Low
   * **Component:** Specify system modules (e.g., "Login Page")
   * **Estimated Time:** Time needed to execute the test
   * **Labels:** Categorization tags (e.g., "Automation")

**Example Test Case**

| **Field** | **Value** |
| --- | --- |
| Name | Verify login with valid email |
| Precondition | User is registered |
| Status | Draft |
| Priority | High |
| Component | Login Page |
| Labels | Automation |

**4. Writing a Test Script**

Test scripts define the step-by-step process to execute the test case.

**Types of Test Scripts in Zephyr Scale:**

1. **Plain Text** – A simple text description.
2. **Step-by-Step** – Detailed steps with expected results (preferred method).
3. **Gherkin (BDD Format)** – Behavior-driven development test scenarios.

**Example of a Step-by-Step Test Script**

| **Step** | **Action** | **Test Data** | **Expected Result** |
| --- | --- | --- | --- |
| 1 | Open login page | N/A | Login page is displayed |
| 2 | Enter valid username | "[user@test.com](mailto:user@test.com)" | Username field is populated |
| 3 | Enter valid password | "password123" | Password field is populated |
| 4 | Click "Login" button | N/A | User is redirected to homepage |

**5. Organizing Test Cases**

Test cases can be structured using folders for better management.

**Folder Organization Approaches:**

1. **Based on Functionality:**
   * Login
   * Signup
   * Cart
2. **Based on Testing Type:**
   * Regression Testing
   * Smoke Testing
   * Performance Testing

**Example Folder Structure:**

Project

│── Test Cases

│ ├── Login

│ │ ├── Verify login with valid credentials

│ │ ├── Verify login with invalid password

│ ├── Signup

│ │ ├── Verify successful signup

│ ├── Cart

│ │ ├── Verify adding items to cart

**6. Test Case Execution**

Once test cases are created, they can be executed as part of a test cycle.

**Execution Process:**

1. **Select a Test Cycle** → Add test cases
2. **Execute Test Cases** → Mark as Passed, Failed, or Blocked
3. **Attach Screenshots & Comments**
4. **Generate Reports** for tracking progress

**Example Execution Report**

| **Test Case** | **Status** | **Execution Time** | **Comments** |
| --- | --- | --- | --- |
| Verify login with valid email | Passed | 5 mins | Works as expected |
| Verify login with invalid email | Failed | 4 mins | Incorrect error message |
| Verify adding items to cart | Passed | 3 mins | No issues found |

**7. Traceability & Reporting**

Zephyr Scale provides traceability by linking test cases to Jira issues (e.g., user stories and defects).

**Example Traceability:**

* **User Story:** "As a user, I want to log in"
* **Linked Test Cases:**
  + Verify login with valid email
  + Verify login with invalid password

**Diagram: Test Case Traceability**

+-----------------------+

| Jira User Story |

| "As a user, I want |

| to log in" |

+-----------------------+

|

|

+-----------------------+

| Linked Test Cases |

| - Verify login |

| - Verify invalid login |

+-----------------------+

**Reports**

* **Test Execution Report** – Displays the status of test cases.
* **Test Coverage Report** – Shows the percentage of test cases executed.

**8. Conclusion**

Zephyr Scale helps teams manage test cases efficiently in Jira. By organizing test cases, executing them in cycles, and linking them to requirements, teams can improve software quality and traceability.

This guide provides a foundational understanding of Zephyr Scale. As you gain experience, explore automation and integrations to enhance test management further.

**Introduction to Zephyr Scale and Test Case Creation**

**What is Zephyr Scale?**

Zephyr Scale is a test management solution that integrates with Jira, a popular project management tool. It allows teams to create, manage, and execute test cases efficiently. With Zephyr Scale, testers can organize test cases into folders, track test execution results, and generate reports for better insights.

**Key Features of Zephyr Scale:**

* **Test Case Management**: Organize and manage test cases within Jira.
* **Test Execution**: Track test runs and results.
* **Integration with Jira**: Link test cases with user stories, bugs, and issues.
* **Reporting**: Generate test execution reports and metrics.
* **Collaboration**: Enables team collaboration on test planning and execution.

**Understanding Test Cases**

A test case is a set of conditions or actions to verify a specific functionality of an application. It includes:

* **Test Case ID**: Unique identifier for the test case.
* **Test Summary**: Brief description of what the test case covers.
* **Preconditions**: Any setup required before executing the test.
* **Test Steps**: Step-by-step instructions to perform the test.
* **Expected Result**: The expected outcome after execution.
* **Actual Result**: The actual outcome observed.
* **Status**: Whether the test passed, failed, or is blocked.

**Example of a Test Case:**

| **Test Case ID** | **TC001** |
| --- | --- |
| Summary | Verify user can sign up |
| Preconditions | User must have an email address |
| Test Steps | 1. Open the application 2. Navigate to the sign-up page 3. Enter email and password 4. Click 'Sign Up' |
| Expected Result | User should be successfully registered |
| Actual Result | (To be filled after execution) |
| Status | Pass/Fail |

**Budget Manager Application Overview**

The Budget Manager application consists of different modules, each requiring specific test cases. The modules include:

1. **Sign Up** - User registration process.
2. **Login** - Authentication system.
3. **Password Reset** - Allows users to recover their password.
4. **Settings** - Includes currency selection and rating functionality.
5. **Card & Cash** - Tracks user expenses and income.
6. **Statistics** - Displays financial insights.
7. **Savings** - Helps users set and track saving goals.

**Organizing Test Cases in Zephyr Scale**

Zephyr Scale allows structuring test cases in a hierarchical manner using folders. The Budget Manager application's test cases can be categorized as follows:

Budget Manager Test Cases

├── Login

│ ├── TC101 - Verify successful login

│ ├── TC102 - Verify incorrect password behavior

├── Sign Up

│ ├── TC201 - Verify successful registration

│ ├── TC202 - Verify invalid email handling

├── Settings

│ ├── TC301 - Verify currency change

│ ├── TC302 - Verify 'Rate Us' redirection

├── Card & Cash

│ ├── TC401 - Verify income addition

│ ├── TC402 - Verify expense withdrawal

│ ├── Statistics

│ │ ├── TC501 - Verify monthly expense report

├── Savings

│ ├── TC601 - Verify savings goal creation

**Writing Test Cases in Zephyr Scale**

1. **Open Zephyr Scale in Jira**.
2. **Navigate to the Test Case Section**.
3. **Create a New Test Case**.
4. **Enter Test Case Details**.
5. **Save and Organize the Test Case into the Correct Folder**.
6. **Execute the Test Case and Update the Status**.

**Conclusion**

Zephyr Scale simplifies test case management by offering structured test organization, seamless integration with Jira, and detailed reporting capabilities. By following the structured approach outlined above, teams can ensure efficient test coverage for their applications.

**Study Material: Writing Test Cases in Zephyr Scale**

**Introduction to Zephyr Scale**

Zephyr Scale is a test management solution integrated with Jira that helps teams manage test cases, test plans, and executions efficiently. It provides structured test case organization and reporting capabilities to streamline the testing process.

**Writing Test Cases in Zephyr Scale**

**1. Understanding Test Cases**

A test case is a set of conditions or actions executed to verify that a system works as expected. It typically includes:

* **Test Case Title**: A brief name describing the test.
* **Description**: Explanation of what the test case does.
* **Preconditions**: Conditions that must be met before executing the test.
* **Test Steps**: Step-by-step instructions to perform the test.
* **Expected Result**: The expected outcome of executing the test.
* **Status & Priority**: To track test execution progress and importance.

**2. Strategy for Writing Test Cases**

1. **List All Scenarios**: Identify all possible scenarios before detailing steps.
2. **Write Titles First**: Capture the essence of the test case in the title.
3. **Add Test Steps Later**: Elaborate steps if needed, depending on team size.
4. **Adjust Based on Team**: If testing solo, minimal detail is required; if working in a team, detailed steps help collaboration.

**3. Example: Sign-Up Functionality**

Consider a sign-up form with the fields:

* **Email**
* **Password**
* **Confirm Password**

| **Test Case ID** | **Test Case Title** | **Steps** | **Expected Result** |
| --- | --- | --- | --- |
| TC001 | Verify sign-up using a valid email | 1. Enter a valid email 2. Enter a strong password 3. Confirm password 4. Click Sign Up | User should successfully sign up |
| TC002 | Verify sign-up with an empty email field | 1. Leave email field blank 2. Enter password & confirm password 3. Click Sign Up | Error message should appear |
| TC003 | Verify sign-up with an empty password field | 1. Enter email 2. Leave password field blank 3. Click Sign Up | Error message should appear |
| TC004 | Verify sign-up with an empty confirm password field | 1. Enter email 2. Enter password 3. Leave confirm password blank 4. Click Sign Up | Error message should appear |
| TC005 | Verify sign-up with mismatched passwords | 1. Enter email 2. Enter different passwords in both fields 3. Click Sign Up | Error message should appear |
| TC006 | Verify Show Password feature | 1. Enter password 2. Click "Show Password" | Password should be visible |
| TC007 | Verify navigation to the Login page | 1. Click on "Already have an account? Login" | User should be redirected to the login page |
| TC008 | Verify sign-up with an invalid email format | 1. Enter an invalid email (e.g., "test@com") 2. Enter password & confirm password 3. Click Sign Up | Error message should appear |
| TC009 | Verify sign-up with an already registered email | 1. Enter an email already used 2. Enter password & confirm password 3. Click Sign Up | Error message should appear |
| TC010 | Verify sign-up with an easy-to-guess password | 1. Enter email 2. Enter simple password (e.g., "123456") 3. Click Sign Up | Error message should appear |
| TC011 | Verify sign-up with a short password | 1. Enter email 2. Enter a short password (e.g., "abc") 3. Click Sign Up | Error message should appear |

**4. Diagram: Workflow of Test Case Execution**

+-------------------+

| Identify Scenarios |

+-------------------+

|

v

+----------------------+

| Write Test Case Titles |

+----------------------+

|

v

+----------------------+

| Add Test Steps |

+----------------------+

|

v

+----------------------+

| Execute and Validate |

+----------------------+

|

v

+----------------------+

| Report Results |

+----------------------+

**Conclusion**

Writing test cases in Zephyr Scale follows a structured approach that helps in effective test management. Identifying all possible scenarios and systematically documenting them ensures thorough test coverage and streamlined testing processes.

**Study Material: Login Test Cases**

**Introduction**

Testing the login functionality of an application ensures that users can securely access their accounts with valid credentials and are appropriately restricted when using invalid credentials. This document explains the fundamental concepts of login test cases in a simple and easy-to-understand manner.

**What is Login Testing?**

Login testing is the process of verifying the authentication system of an application. It ensures that only authorized users can access the application using their credentials (email and password) and that invalid attempts are handled correctly.

**Elements in a Login Screen**

A typical login screen consists of:

* **Email/Username field** - User enters their registered email or username.
* **Password field** - User enters their account password.
* **Login button** - Submits the login form for authentication.
* **Forgot Password link** - Allows users to reset their password if forgotten.
* **Sign Up link** - Redirects users to the registration page.
* **Show Password option** - Allows users to see the entered password.

**Test Cases for Login Functionality**

| **Test Case ID** | **Description** | **Expected Outcome** |
| --- | --- | --- |
| TC01 | Verify login with a valid email and valid password | User should be able to log in successfully |
| TC02 | Verify login with an invalid email | Error message should appear: "Invalid email format" |
| TC03 | Verify login with an empty email field | Error message should appear: "Email cannot be empty" |
| TC04 | Verify login with an empty password field | Error message should appear: "Password cannot be empty" |
| TC05 | Verify navigating from login page to sign-up page | User should be redirected to the sign-up page |
| TC06 | Verify navigating from login page to forgot password page | User should be redirected to the forgot password page |
| TC07 | Verify show password functionality | The password should be displayed in plain text when enabled |
| TC08 | Verify login screen functionality in portrait mode | The login screen should be displayed correctly in portrait mode |
| TC09 | Verify login screen functionality in landscape mode | The login screen should be displayed correctly in landscape mode |

**Explanation with Examples**

**1. Valid Login Test**

**Scenario:**

* User enters a valid email: user@example.com
* User enters a valid password: SecurePass123
* Clicks on the login button.

**Expected Result:**

* User should be logged in and redirected to the home page.

**2. Invalid Email Format**

**Scenario:**

* User enters an invalid email: userexample.com (missing @ symbol)
* Clicks on the login button.

**Expected Result:**

* An error message should appear: "Invalid email format. Please enter a valid email."

**3. Empty Password Field**

**Scenario:**

* User enters a valid email but leaves the password field empty.
* Clicks on the login button.

**Expected Result:**

* An error message should appear: "Password cannot be empty."

**4. Navigation Test: Login to Forgot Password Page**

**Scenario:**

* User clicks on the "Forgot Password" link on the login screen.

**Expected Result:**

* User should be redirected to the forgot password page.

**Diagram: Login Process Flow**

+------------------+

| Open Login Page |

+------------------+

|

v

+------------------+

| Enter Email & Pwd|

+------------------+

|

v

+------------------+

| Click Login |

+------------------+

| |

| Success | Failure

v v

+------------------+ +------------------+

| Redirect to Home | | Show Error Msg |

+------------------+ +------------------+

**Conclusion**

Login test cases ensure that the authentication mechanism works correctly and securely. By testing different scenarios, including valid and invalid inputs, UI elements, and navigation, we can improve the login experience for users and prevent security vulnerabilities.

This study material provides a basic understanding of how to test login functionality in a simple and structured way. 🚀

**Study Material: Forgot Password Functionality**

**1. Introduction**

The **"Forgot Password"** feature is a common functionality in applications that allows users to reset their password if they forget it. This process is crucial for user accessibility and security.

**1.1 What is "Forgot Password"?**

It is a feature that helps users reset their password by providing an email or phone number linked to their account. A reset link or code is sent to the provided contact, allowing them to set a new password.

**2. How "Forgot Password" Works**

Below is a simple flowchart to explain the **Forgot Password** process:

**Step 1:** User clicks on the "Forgot Password" link on the login screen.

**Step 2:** System prompts the user to enter their registered email.

**Step 3:** System verifies if the email is registered.

* If the email exists, a reset password link is sent.
* If the email does not exist, an error message is displayed.

**Step 4:** User receives the reset link and clicks on it.

**Step 5:** User sets a new password and confirms it.

**Step 6:** System updates the password and allows the user to log in with the new credentials.

**3. Test Cases for Forgot Password**

Testing the "Forgot Password" functionality ensures it works as expected. Below are some test cases:

| **Test Case ID** | **Test Scenario** | **Expected Result** |
| --- | --- | --- |
| FP-001 | Verify "Forgot Password" with a valid email | The system sends a reset password email successfully |
| FP-002 | Verify "Forgot Password" with an invalid email | The system displays an error message: "Email not found" |
| FP-003 | Verify "Forgot Password" with an empty email field | The system prompts: "Please enter your email" |
| FP-004 | Verify the reset link expiration | The reset link should expire after a certain period (e.g., 24 hours) |
| FP-005 | Verify password reset with a weak password | The system prompts: "Password must meet security criteria" |
| FP-006 | Verify password reset confirmation | The user should be able to successfully log in with the new password |

**4. Example of Forgot Password Process**

Let’s go through an example scenario of a user trying to reset their password.

**Example: Reset Password for John Doe**

1. John Doe forgets his password and clicks on **"Forgot Password"**.
2. He enters his registered email: [**john.doe@example.com**](mailto:john.doe@example.com).
3. The system verifies the email and sends a reset link.
4. John opens his email and clicks on the **"Reset Password"** link.
5. He enters a new password: **John@12345** and confirms it.
6. The system updates the password and allows him to log in with the new password.

**5. Forgot Password Page UI Layout (Example)**

-----------------------------------

| Forgot Password |

-----------------------------------

| Enter your registered email: |

| [\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_] |

| |

| [ Send Reset Link ] [ Cancel ] |

-----------------------------------

**6. Conclusion**

The "Forgot Password" feature is essential for user convenience and security. Proper testing ensures that users can reset their passwords without issues. Implementing security measures, such as **email verification, password strength validation, and link expiration**, is crucial to prevent unauthorized access.

Let me know if you need any modifications! 🚀

**Basic Study Material: Understanding App Testing Concepts**

**1. Introduction to Settings Testing**

**What is Settings Testing?**

Settings testing ensures that all navigations, functionalities, and modifications in an application's settings section work as expected.

**Example & Explanation**

Consider a mobile application with a "Settings" menu containing the following options:

* **About** (displays app information)
* **Currency** (allows changing currency for transactions)
* **Rate Us** (lets users rate the app)

Each of these settings should be tested to verify proper navigation, changes reflecting in the application, and correct user interactions.

**Test Cases for Settings**

| **Test Case** | **Description** |
| --- | --- |
| Verify navigation | Ensure users can move between settings and other sections |
| Verify "About" page | Open the "About" page and check displayed app info |
| Verify currency change | Change currency and check if the change reflects across the app |
| Verify rating feature | Ensure users can rate the app if it's available in the store |
| Verify landscape mode | Check the settings section in landscape mode |

**2. Card and Cash Transactions Testing**

**What is Transaction Testing?**

Transaction testing involves verifying the flow of adding, withdrawing, and transferring money between different financial sections of an application.

**Example & Explanation**

In a finance management app, users may have a **Card** and **Cash** section where they:

* Deposit money into a card
* Withdraw money from a card
* Transfer money between card and cash
* Check transaction history

**Test Cases for Card & Cash**

| **Test Case** | **Description** |
| --- | --- |
| Verify deposit to card | Add money to a card and check balance update |
| Verify withdrawal from card | Withdraw money and verify the deducted amount |
| Verify deposit to cash | Add money to cash and check balance update |
| Verify withdrawal from cash | Withdraw from cash and check if it reflects correctly |
| Verify transaction history | Ensure transaction logs are properly recorded |
| Verify landscape mode | Check the functionality in landscape mode |

**Diagram Representation:**

(User) → Deposits → (Card) → Transfers → (Cash)

→ Withdraws → (Card)

→ Withdraws → (Cash)

**3. Statistics Testing**

**What is Statistics Testing?**

Statistics testing ensures that financial data over different time periods is correctly displayed to users.

**Example & Explanation**

A finance application may display spending statistics for:

* The last 7 days
* The last month (31 days)
* The last 12 months

**Test Cases for Statistics**

| **Test Case** | **Description** |
| --- | --- |
| Verify 7-day statistics | Ensure correct spending trends are shown for the past week |
| Verify monthly statistics | Check if the monthly financial report is accurate |
| Verify yearly statistics | Validate the financial summary for the past 12 months |
| Verify correct UI | Ensure the statistics section is user-friendly and understandable |

**4. Savings Testing**

**What is Savings Testing?**

Savings testing verifies the ability to create, modify, and track savings within an application.

**Example & Explanation**

Users may create savings goals in a budgeting app. The app should allow:

* Adding a new savings goal
* Editing or deleting a savings goal
* Keeping track of saved and remaining amounts

**Test Cases for Savings**

| **Test Case** | **Description** |
| --- | --- |
| Verify creating savings | Ensure users can successfully create a savings goal |
| Verify empty field validation | Prevent users from saving if required fields are empty |
| Verify valid input | Ensure only numbers are accepted in amount fields |
| Verify editing savings | Modify a savings goal and confirm updates |
| Verify multiple savings | Handle multiple savings goals without errors |
| Verify UI and usability | Check if users can easily navigate savings features |

**Diagram Representation:**

(User) → Creates → (Savings Goal: Buy a Car)

→ Adds Funds → (Updated Savings Amount)

→ Withdraws → (Deducts from Savings)

**Conclusion**

By systematically testing these features, users can ensure the reliability and usability of financial applications. Testing ensures that:

1. **Navigation works correctly** (e.g., moving between settings and transactions)
2. **Transactions reflect accurately** (e.g., deposits, withdrawals, and transfers work as expected)
3. **Statistics are meaningful** (e.g., financial summaries match real-time data)
4. **Savings functionality operates smoothly** (e.g., users can create, edit, and track savings effectively)

**Final Tip:** A tester should not only focus on function but also on usability, ensuring that the application is intuitive for all users!

**Basic Study Material on Software Testing and Test Case Implementation**

**Introduction to Software Testing**

Software testing is the process of evaluating a software application to identify and fix any bugs, errors, or inconsistencies. The goal is to ensure the software performs as expected and meets user requirements.

**1. Understanding Test Cases**

**What is a Test Case?**

A test case is a set of conditions or steps used to check whether a software application functions correctly. Each test case consists of:

* **Preconditions:** The initial state before executing the test.
* **Test Steps:** Actions performed to test the functionality.
* **Expected Results:** The anticipated outcome if the software behaves correctly.

**Example Test Case**

| **Test Case ID** | **TC001** |
| --- | --- |
| **Title** | Login with an Empty Email Field |
| **Precondition** | The login page must be open. |
| **Steps** |  |
| 1. Leave the email field empty. |  |
| 2. Enter a valid password. |  |
| 3. Click on the 'Login' button. |  |
| **Expected Result** | An error message should appear indicating that the email field is required. |
| **Priority** | Low |
| **Status** | Approved |

**2. Test Implementation**

Test implementation involves organizing test cases, defining priorities, structuring test cycles, and estimating execution time.

**Key Components of Test Implementation**

1. **Test Cycles:** A group of test cases executed together.
2. **Test Suites vs. Test Cycles:**
   * **Test Suite:** A collection of test cases that are logically grouped.
   * **Test Cycle:** A scheduled execution of test cases.
3. **Priority Levels:**
   * **High:** Critical tests that must pass.
   * **Normal:** Tests with moderate importance.
   * **Low:** Tests that are less critical.

**Example of Priority Assignment**

| **Test Case** | **Priority** |
| --- | --- |
| Login with valid credentials | High |
| Login with an empty email | Low |
| Navigating from login to forgot password | High |

**3. Zephyr Scale vs. Other Tools**

* Some tools like **TestLink** use the term "Test Suite."
* **Zephyr Scale** replaces test suites with folders and test cycles.

**Diagram: Test Case Structure in Zephyr Scale**

Test Cases --> Folders --> Test Cycles

**4. Test Case Execution and Review**

**Who Reviews Test Cases?**

In some companies, one tester writes test cases, while another tester reviews them before approval.

**Example: Reviewing and Approving Test Cases**

| **Test Case** | **Status** |
| --- | --- |
| Login with invalid email | Approved |
| Forgot password with an empty email | Approved |

**5. Expected Results and Error Messages**

A good error message should be clear and specific.

**Example:**

* **Bad Error Message:** "Something went wrong."
* **Good Error Message:** "Email field cannot be empty. Please enter your email."

**Diagram: Expected Error Message Display**

+-----------------------+

| ERROR: Email field |

| cannot be empty. |

+-----------------------+

**Conclusion**

* Understanding test cases helps in ensuring software reliability.
* Prioritizing test cases ensures critical functionalities are tested first.
* Zephyr Scale organizes test cases differently than other tools.
* Well-defined expected results improve user experience.

By following these concepts, beginners can start learning software testing in a structured manner.

**Study Material: Understanding Test Cases and Test Cycles**

**1. Introduction to Test Cases**

A **test case** is a set of steps and conditions used to check whether a software application is working correctly. It includes:

* **Priority**: How important the test is.
* **Status**: Whether it is executed or not.
* **Steps**: Actions to perform.
* **Expected Result**: What should happen after performing the steps.
* **Data**: Any input needed for testing.

**Example:**

| **Test Case ID** | **Description** | **Steps** | **Expected Result** |
| --- | --- | --- | --- |
| TC\_001 | Login Test | Enter username & password, click Login | User should log in successfully |
| TC\_002 | Invalid Login Test | Enter wrong credentials, click Login | Error message should appear |

**2. What is a Test Cycle?**

A **test cycle** is a group of test cases that are executed together. It helps to manage testing efficiently.

**Types of Test Cycles:**

1. **Smoke Testing** – A quick test to check if the main features work.
2. **Regression Testing** – Tests all functionalities to ensure nothing is broken after changes.
3. **Full Regression Testing** – More extensive testing covering all possible scenarios.

**Example:**

| **Test Cycle Name** | **Purpose** | **When to Run?** |
| --- | --- | --- |
| Smoke Testing | Check basic functionality | Every new release |
| Regression Test | Verify recent changes didn't break anything | Before major releases |
| Full Regression Test | Comprehensive testing of the entire system | Monthly or after major updates |

**3. Running a Test Cycle**

To execute a test cycle:

1. **Create a new test cycle**
2. **Assign test cases**
3. **Execute test cases and track status**
4. **Report defects if needed**
5. **Save test execution results**

**Test Execution Status:**

* **Not Executed**: Test not yet started.
* **In Progress**: Test is being run.
* **Passed**: Test was successful.
* **Failed**: Test did not meet expected results.
* **Blocked**: Cannot proceed due to an issue.

**Example:**

| **Test Case ID** | **Status** | **Time Taken** |
| --- | --- | --- |
| TC\_001 | Passed | 5 min |
| TC\_002 | Failed | 3 min |
| TC\_003 | Blocked | - |

**4. Test Estimation and Tracking Progress**

Estimating time for each test case helps in planning the test execution.

**Why is Estimation Important?**

* Helps in project planning.
* Ensures enough time is allocated.
* Avoids delays in software release.

**Example:**

If a cycle has 10 test cases, each taking 10 minutes, total estimated time = **10 x 10 = 100 minutes**.

**Tracking Progress:**

* Use timers to track actual test execution time.
* Compare estimated vs. actual time.
* Adjust test plans based on findings.

**5. Conclusion**

Understanding test cases and test cycles is essential in software testing. Organizing test cases efficiently and running structured test cycles ensures software reliability and quality.

By following proper test execution steps and tracking progress, teams can identify defects early and improve the software development process.

**Study Material for Basic Level Users**

**Introduction to Test Reports in Zephyr Scale**

**What are Test Reports?**

Test reports provide insights into test execution, results, and overall testing progress. They help teams understand the status of software testing, identify issues, and make informed decisions.

**Types of Test Reports**

**1. Test Execution Reports**

These reports show the results of executed test cases within a test cycle.

**Example:**

A team executed 50 test cases, out of which:

* 30 Passed ✅
* 10 Failed ❌
* 5 Blocked 🚫
* 5 Not Executed ⚪

| **Status** | **Number of Cases** |
| --- | --- |
| Passed | 30 |
| Failed | 10 |
| Blocked | 5 |
| Not Executed | 5 |

**2. Test Progress Summary**

This report provides an overview of testing progress over time.

**Example:**

A team planned to run 100 test cases over 10 days. On day 5, they completed 40 tests.

| **Day** | **Test Cases Executed** |
| --- | --- |
| 1 | 10 |
| 2 | 10 |
| 3 | 5 |
| 4 | 10 |
| 5 | 5 |
| 6-10 | Remaining |

**3. Test Coverage Report**

This report shows how many test cases cover each requirement or user story.

**Example:**

A project has 5 user stories, and each story should have at least 2 test cases.

| **User Story** | **Test Cases Linked** |
| --- | --- |
| Login Feature | 3 |
| Payment Process | 4 |
| Dashboard | 2 |
| Profile Update | 3 |
| Logout | 1 |

If any requirement has **0 test cases**, it indicates incomplete testing.

**Key Features in Zephyr Scale Reports**

1. **Filtering by Date:**
   * View reports for a specific date or time period.
   * Example: Show test executions completed on **16th February**.
2. **Test Cycles:**
   * A test cycle is a group of test cases executed together.
   * Example: A cycle named **"Smoke Testing"** runs before the **"Regression Testing"** cycle.
3. **Tester-wise Reports:**
   * Shows individual tester contributions.
   * Example: Tester **John** executed 20 test cases, **Alice** executed 15 test cases.
4. **Estimated vs Actual Time:**
   * Helps in tracking testing efficiency.
   * Example: Estimated time for execution **= 11 minutes**, Actual time **not recorded**.

**Conclusion**

Test reports in Zephyr Scale provide valuable insights into testing status, efficiency, and coverage. Understanding these reports helps teams improve test management, track defects, and ensure product quality.