**Study Material: Test Scenarios for Beginners**

**1. Introduction to Test Scenarios**

**What is a Test Scenario?**

A **Test Scenario** is a description of a functionality or feature that needs to be tested in a software application. It helps testers ensure that all the major functionalities of an application work as expected.

**Why are Test Scenarios Important?**

* They provide a high-level idea of what to test.
* They help identify real-world use cases of the application.
* They assist in prioritizing test cases and ensure maximum coverage.
* They make it easy to understand and execute testing steps.

**2. Test Scenarios vs Test Cases**

| **Feature** | **Test Scenario** | **Test Case** |
| --- | --- | --- |
| Definition | A broad description of a feature to be tested | Detailed steps to validate a scenario |
| Detail Level | High-level | Low-level (step-by-step) |
| Example | "Verify that a user can log in" | "Step 1: Open the login page, Step 2: Enter credentials, Step 3: Click Login" |
| When to Use | When time is limited, for high-level understanding | When detailed verification is required |

**3. How to Write Test Scenarios**

**Step 1: Study Requirements**

* Read Business Requirement Specification (BRS) or Software Requirement Specification (SRS).
* If no documentation exists, study the UI design, prototypes, or older versions of the application.

**Step 2: Identify Major Functionalities**

* Break down the application into key components (e.g., login, registration, shopping cart).

**Step 3: Define User Flows and Possible Actions**

* Think from the user’s perspective and list common activities.

**Step 4: Write Test Scenarios**

* Use clear and simple language.
* Avoid too much detail (details belong in test cases).

**Step 5: Prioritize and Review**

* Mark critical scenarios to be tested first.
* Get feedback from team members or supervisors.

**4. Example Test Scenarios**

**Example 1: Registration Page Test Scenarios**

| **Scenario ID** | **Test Scenario Description** |
| --- | --- |
| REG-01 | Verify that a user can successfully register with valid details |
| REG-02 | Verify that registration fails if mandatory fields are left blank |
| REG-03 | Verify that an error message appears if the email format is invalid |
| REG-04 | Verify that the password meets security requirements |

**Example 2: Login Page Test Scenarios**

| **Scenario ID** | **Test Scenario Description** |
| --- | --- |
| LOGIN-01 | Verify that a user can successfully log in with valid credentials |
| LOGIN-02 | Verify that login fails with incorrect username or password |
| LOGIN-03 | Verify that the system locks the account after multiple failed attempts |
| LOGIN-04 | Verify that the "Forgot Password" functionality works correctly |

**5. Tools for Writing Test Scenarios**

**Common Tools Used:**

| **Tool** | **Type** |
| --- | --- |
| Trello | Task Management |
| Excel/Google Sheets | Manual Documentation |
| Jira | Test Case Management |
| TestRail | Test Management |

* Companies often have their preferred tools, but test scenarios can be written using any documentation tool.
* The key skill is understanding how to create **effective test scenarios**, not just knowing how to use a tool.

**6. Conclusion**

* Writing test scenarios is essential for systematic software testing.
* They provide a **structured approach** to validating functionalities.
* Prioritizing test scenarios ensures that **critical features are tested first**.
* Tools like **Trello, Excel, or Jira** help organize and track test scenarios effectively.

In the next steps, we will sign up for **Trello**, create a test board, and start writing our first test scenario!

**Study Material: Basic Introduction to Trello**

**What is Trello?**

Trello is a web-based project management tool that helps individuals and teams organize tasks, collaborate, and improve productivity. It uses a visual interface based on boards, lists, and cards to help users track and manage their work effectively.

**Example:**

* A software development team uses Trello to track the progress of their tasks.
* A student uses Trello to manage assignments and deadlines.

**Key Features of Trello**

| **Feature** | **Description** |
| --- | --- |
| **Boards** | Represents a project or a workspace where tasks are managed. |
| **Lists** | Used to organize tasks within a board. Example: "To Do," "In Progress," "Completed." |
| **Cards** | Individual tasks or items placed in lists. These can contain descriptions, attachments, due dates, and labels. |
| **Workspaces** | A collection of boards, typically used for a team or department. |
| **Collaboration** | Users can add team members, assign tasks, and communicate through comments. |
| **Automation (Butler)** | Automates repetitive tasks within Trello. |
| **Integration** | Connects with tools like Slack, Google Drive, and Jira. |

**Getting Started with Trello**

**Step 1: Sign Up**

1. Go to [Trello's Website](https://trello.com).
2. Click "Sign Up" and enter your email.
3. Confirm your email and create a password.
4. Login using your Atlassian account (if available).

**Step 2: Creating a Workspace**

1. Click "Create a new workspace."
2. Enter a workspace name (e.g., "Testing Bootcamp").
3. Invite team members (optional).

**Step 3: Creating a Board**

1. Inside the workspace, click "Create a board."
2. Name the board (e.g., "Project Management").
3. Set visibility:
   * **Private:** Only invited members can see it.
   * **Workspace Visible:** All members of the workspace can see it.
   * **Public:** Anyone on the internet can view it (Not recommended for confidential projects).

**Step 4: Adding Lists and Cards**

1. Create lists such as:
   * **To Do**
   * **In Progress**
   * **Completed**
2. Add cards under each list to represent individual tasks.
3. Click on a card to add:
   * Task details
   * Attachments
   * Due dates
   * Labels (e.g., "Urgent," "Bug Fix")
   * Assigned team members

**Example: Using Trello for Software Testing**

**Scenario: Managing Test Cases**

1. **Board:** "Software Testing"
2. **Lists:**
   * Test Case Design
   * Test Execution
   * Defects Found
   * Completed Tests
3. **Cards:**
   * "Login Functionality Testing"
   * "Checkout Process Testing"
   * "Database Performance Testing"
4. Assign each card to a tester, set due dates, and track progress.

**Diagram:**

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

| Software Testing |

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

| Test Case Design | Test Execution | Defects Found | Completed Tests |

|-----------------|-----------------|--------------|----------------|

| Card: Login Test | Card: Checkout Test | Card: Bug #101 | Card: UI Fix |

| Assigned: Tester A | Assigned: Tester B | Assigned: Dev Team | Assigned: Tester C |

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

**Best Practices for Using Trello**

1. **Keep Boards Organized:** Use clear names and structured lists.
2. **Use Labels & Filters:** Color-coded labels for priority tasks.
3. **Set Due Dates:** Helps manage deadlines efficiently.
4. **Collaborate Effectively:** Use comments to communicate with team members.
5. **Use Automation:** Automate task movements and notifications.
6. **Regularly Review Boards:** Ensure tasks are up-to-date.

**Conclusion**

Trello is an easy-to-use and highly flexible tool for project and task management. Whether you are a developer, tester, student, or freelancer, Trello helps streamline workflows and improve collaboration. By understanding boards, lists, and cards, you can manage your tasks efficiently and enhance productivity.

**Further Learning**

* Explore Trello's official documentation.
* Try different templates for different project types.
* Experiment with automation using Trello’s Butler feature.

**Happy Task Management with Trello!** 🎯

**Study Material: Basic Understanding of Test Scenarios and Functional Requirements**

**1. Introduction to Test Scenarios**

Test scenarios are descriptions of how a user might interact with an application to ensure it functions correctly. They help testers identify various cases to validate the software's behavior.

**2. Understanding Functional and Non-Functional Requirements**

**2.1 Functional Requirements (FR)**

These are specific functionalities that a system or application must perform. For example:

* A user should be able to register using a username, password, and email.
* The system should allow login using credentials.

**2.2 Non-Functional Requirements (NFR)**

These define system attributes like performance, usability, and security. For example:

* The application should load within 2 seconds.
* User passwords must be encrypted.

**3. Example of a Functional Requirement Document**

| **ID** | **Requirement Description** | **Dependency** |
| --- | --- | --- |
| FR3 | User should be able to register through the mobile app. | FR1 (Download App) |

**Key Fields in a Requirement Document:**

* **ID:** Unique identifier (e.g., FR3).
* **Title:** Brief description (e.g., User Registration - Mobile Application).
* **Description:** Details of the feature (e.g., username, password, email required).
* **Rationale:** Why is this requirement needed?
* **Dependency:** Specifies the sequence of execution.

**4. Writing Test Scenarios**

Test scenarios can be written in three perspectives:

* **First-person:** "I enter the username."
* **Second-person:** "You enter the username."
* **Third-person:** "The user enters the username." (Preferred approach)

**5. Types of Test Scenarios**

Test scenarios are classified as:

1. **Positive (Valid) Scenarios** – Expected actions work correctly.
2. **Negative (Invalid) Scenarios** – System handles errors properly.

**5.1 Positive Scenarios for Registration**

| **Scenario ID** | **Description** |
| --- | --- |
| TS1 | Verify registration using valid username, password, and email. |
| TS2 | Verify registration using valid username, password, email, and phone number. |

**5.2 Negative Scenarios for Registration**

| **Scenario ID** | **Description** |
| --- | --- |
| TS3 | Verify registration while leaving the username field empty. |
| TS4 | Verify registration while leaving the password field empty. |
| TS5 | Verify registration while leaving the email field empty. |
| TS6 | Verify registration using a very short username (e.g., "hi"). |
| TS7 | Verify registration using an already existing username. |
| TS8 | Verify registration using a non-English username. |

**6. Testing Password Restrictions**

**6.1 Common Password Rules**

| **Rule No.** | **Restriction** |
| --- | --- |
| 1 | Password must be at least 6 characters long. |
| 2 | Password must include uppercase and lowercase letters. |
| 3 | Password must be alphanumeric (letters & numbers). |

**6.2 Negative Scenarios for Password**

| **Scenario ID** | **Description** |
| --- | --- |
| TS9 | Verify registration with a password shorter than 6 characters. |
| TS10 | Verify registration with a password without uppercase letters. |
| TS11 | Verify registration with a password without lowercase letters. |
| TS12 | Verify registration with a password that does not contain numbers. |

**7. Importance of Testing Each Field Separately**

When testing, it is crucial to:

* Make an error in only one field at a time.
* Keep other fields valid to identify the exact cause of the issue.

**8. Conclusion**

Test scenarios help ensure an application functions correctly by covering valid and invalid cases. Writing clear and structured scenarios enhances the efficiency of testing and helps detect issues early.

**Basic Study Material on Mobile App Testing**

**1. Understanding Mobile App Sign-Up and OTP Authentication**

**What is Sign-Up and Why is it Important?**

Sign-up is the process of creating an account in an application by providing personal details like phone number, email, or social media credentials. It is essential for:

* Personalizing user experience.
* Enabling secure access to app features.
* Storing user preferences and history.

**Example: KFC Mobile App Sign-Up**

In the KFC mobile app, users register using their **phone number** instead of an **email address**. This method is common in delivery applications because:

* Delivery personnel need to contact users directly.
* Email notifications are not instant for real-time communication.

**Comparison of Sign-Up Methods**

| **Feature** | **Email Sign-Up** | **Phone Number Sign-Up** |
| --- | --- | --- |
| **Recovery Method** | Email | OTP via SMS |
| **Required for Registration?** | Optional | Mandatory |
| **Usability** | Can be used for marketing emails | Used for instant communication |

**2. Understanding OTP (One-Time Password) Authentication**

OTP is a **temporary code** sent to the user's phone via SMS, valid for a short duration. It is used to verify user identity before accessing the app.

**OTP Workflow in Mobile Applications**

1. User enters their phone number.
2. System sends an OTP via SMS.
3. User enters the OTP in the app.
4. System verifies the OTP and grants access.

**Example: Uber Login**

* When logging in, Uber users enter their phone number.
* They receive a 4-digit OTP via SMS.
* Entering the correct OTP grants access.

**Why is OTP Important?**

* **Enhances Security**: Prevents unauthorized access.
* **Quick Verification**: Eliminates the need for remembering passwords.
* **Ensures Valid Phone Numbers**: Prevents fake account creation.

**3. Testing Sign-Up and OTP Scenarios**

**A. Sign-Up Scenarios**

| **Scenario** | **Expected Outcome** |
| --- | --- |
| Entering a valid phone number | Account is created successfully |
| Entering an invalid phone number | Error message appears |
| Entering characters instead of numbers | Error message appears |
| Entering an Arabic phone number | Accepted if supported |

**B. OTP Verification Scenarios**

| **Scenario** | **Expected Outcome** |
| --- | --- |
| Entering correct OTP | Successful login |
| Entering incorrect OTP | Error message appears |
| Entering expired OTP | Error message appears |
| Entering OTP in different language format | Accepted if supported |

**4. Additional Mobile App Testing Considerations**

**A. Screen Rotation**

* Apps should function correctly in both **portrait** and **landscape** modes.

**B. Numeric Keypad for Phone Number Input**

* When clicking on the phone number field, the **numeric keypad** should appear instead of a QWERTY keyboard.

**C. Localization Testing**

* Phone number formats differ by country (e.g., Egypt’s country code +20, omitting leading 0s).
* OTP messages should be sent in the user's preferred language.

**Conclusion**

Sign-up and OTP authentication are **critical components** of mobile applications, especially in delivery services. Proper testing ensures:

* **Secure and smooth registration**.
* **Seamless user experience** across different countries and languages.
* **Prevention of fraud** through valid phone number verification.

By following structured testing scenarios, we can ensure that users can **register, authenticate, and access mobile applications reliably and securely.**

**Basic Study Material: Understanding Test Scenarios for Sign-Up Functionality**

**Introduction to Test Scenarios**

A test scenario is a set of actions that help validate the functionality of an application. It helps ensure that a feature works as expected and identifies potential issues.

For example, when testing the **sign-up functionality** of a website like Facebook, we need to verify various fields such as First Name, Last Name, Mobile Number, Email, Password, Date of Birth, and Gender.

**Understanding the Sign-Up Page**

The Facebook sign-up page consists of six main fields:

1. **First Name**
2. **Last Name (Surname)**
3. **Mobile Number or Email Address**
4. **New Password**
5. **Date of Birth**
6. **Gender**

Each of these fields must be tested under different conditions to ensure smooth functionality.

**Basic Test Scenarios for Sign-Up Page**

To verify sign-up functionality, we must create both valid and invalid test scenarios. Below are the common test cases:

**1. Valid Test Scenarios**

| **Test Case** | **Expected Outcome** |
| --- | --- |
| Register with a valid phone number | User successfully registered |
| Register with a valid email address | User successfully registered |
| Register with Gender as Male | User successfully registered |
| Register with Gender as Female | User successfully registered |
| Register with Gender as Custom | User successfully registered |

**2. Invalid Test Scenarios**

| **Test Case** | **Expected Outcome** |
| --- | --- |
| Leaving First Name field empty | Error message displayed |
| Leaving Last Name field empty | Error message displayed |
| Providing only one character in First Name | Error message displayed |
| Providing numbers or special characters in Name fields | Error message displayed |
| Providing invalid email format (e.g., abc@com) | Error message displayed |
| Entering less than the required number of characters in Password | Error message displayed |

**Understanding the Restrictions on Fields**

**First Name & Last Name:**

* Cannot contain numbers or special characters.
* Can be in any language.
* Minimum length is 2 characters.
* Can be duplicated (two people can have the same name).

**Username (Email/Mobile Number):**

* Email must follow proper format (e.g., [name@example.com](mailto:name@example.com)).
* Mobile numbers should be in a valid format and not already registered.

**Password:**

* Should have a minimum length (e.g., 6-8 characters).
* Should contain a mix of letters, numbers, and special characters for security.

**Date of Birth:**

* Selected from a dropdown list to avoid errors.
* Must ensure users above a minimum age requirement can register.

**Gender:**

* Chosen from predefined options (Male, Female, Custom).
* Only one option can be selected at a time.

**Concept of Equivalence Partitioning**

Equivalence partitioning is a technique used in testing to divide input data into valid and invalid groups to reduce test cases while maintaining efficiency.

For example, when testing the **Date of Birth** field:

* **Valid Partition:** Age above 18 years.
* **Invalid Partition:** Age below 18 years.

Using this method, we ensure all critical scenarios are tested without unnecessary repetition.

**Exhaustive Testing in Gender Field**

Some fields have limited options, allowing us to perform **exhaustive testing**, meaning all possible choices are tested. The Gender field, which has three options, is an example where this approach is feasible.

| **Test Case** | **Expected Outcome** |
| --- | --- |
| Selecting Male | Successfully registered |
| Selecting Female | Successfully registered |
| Selecting Custom | Successfully registered |
| Not selecting any option | Error message displayed |

However, fields like First Name or Password, which have infinite possibilities, cannot be tested exhaustively.

**Conclusion**

Testing the **sign-up functionality** of an application involves identifying potential issues through a variety of test cases. By covering different scenarios, we ensure the system is secure, user-friendly, and error-free. Using techniques like **equivalence partitioning** and **exhaustive testing** improves test coverage and efficiency.

By following these structured test scenarios, we can verify that the sign-up process works correctly and provides a smooth experience for users.

**Study Material for Basic Users**

**1. Importance of Input Validation**

When users enter information in an application (like a website or mobile app), it is important to ensure that the data is correct. Input validation helps prevent errors and ensures that only valid data is accepted.

**Example of Input Validation**

* A mobile number should contain only numbers and should have the correct length.
* An email address should follow the correct format (e.g., [example@domain.com](mailto:example@domain.com)).
* A password should meet security requirements (e.g., at least 8 characters, containing uppercase, lowercase, numbers, and special characters).

**2. Testing Different Input Scenarios**

**A. Mobile Number & Email Address**

When filling out a form, users may enter valid or invalid data. Below are different scenarios for testing:

| **Scenario** | **Expected Outcome** |
| --- | --- |
| Leaving phone number or email empty | Shows an error message: "Field cannot be empty" |
| Entering an invalid phone number (e.g., "123") | Shows an error message: "Invalid phone number" |
| Entering special characters in phone number (e.g., "@#$%^&") | Shows an error message: "Only numbers are allowed" |
| Entering an invalid email format (e.g., "example.com") | Shows an error message: "Invalid email format" |
| Entering an already used email | Shows an error message: "Email is already registered" |

**B. Password Requirements**

A strong password is necessary for security. Below are different password scenarios:

| **Scenario** | **Expected Outcome** |
| --- | --- |
| Leaving password empty | Shows an error: "Password is required" |
| Using a short password (e.g., "123") | Shows an error: "Password must be at least 8 characters" |
| Password without uppercase letters (e.g., "password1") | Shows an error: "Include at least one uppercase letter" |
| Password without lowercase letters (e.g., "PASSWORD1") | Shows an error: "Include at least one lowercase letter" |
| Password without numbers (e.g., "Password!") | Shows an error: "Include at least one number" |
| Password without special characters (e.g., "Password1") | Shows an error: "Include at least one special character" |

**3. Date of Birth (DOB) Validation**

Some applications require users to enter their date of birth. It is important to ensure that users meet the age requirements.

**Example: Age Validation for Facebook**

Facebook allows users between **13 to 120 years** old to register. Any invalid age entry should show an error message.

| **Scenario** | **Expected Outcome** |
| --- | --- |
| Entering a future date | Shows an error: "Date of birth cannot be in the future" |
| Entering an age below 13 | Shows an error: "You must be at least 13 years old to register" |
| Entering an age above 120 | Shows an error: "Please enter a valid age" |

**Diagram: Age Validation Flow**

User enters Date of Birth

|

v

Is the age between 13 and 120?

/ \

Yes No

| |

Proceed Show error message

**4. Gender Selection**

Most registration forms require users to select their gender. This is usually provided as radio buttons (Male, Female, Other). The system should validate that a selection is made.

| **Scenario** | **Expected Outcome** |
| --- | --- |
| No gender selected | Shows an error: "Please select your gender" |
| Valid gender selected | Allows form submission |

**Diagram: Gender Selection in Forms**

[ ] Male [ ] Female [ ] Other (User must select one)

**5. Summary of Best Practices**

To ensure smooth registration and good user experience, applications should follow these best practices:

1. **Require Mandatory Fields**: Ensure essential fields are not left empty.
2. **Use Proper Input Formats**: Validate email, phone numbers, and passwords.
3. **Provide Clear Error Messages**: Help users correct mistakes by showing meaningful errors.
4. **Age Validation**: Ensure users meet the required age for registration.
5. **Secure Passwords**: Enforce strong password rules for better security.

By following these guidelines, applications can improve usability and security, making registration processes smoother and more efficient for users.

**Study Material: Login Functionality & Test Scenarios**

**1. Introduction to Login Functionality**

Login functionality is an essential part of most applications. It allows users to access their accounts securely by verifying their credentials, typically using a combination of:

* **Username & Password**
* **Email & Password**
* **Phone Number & Password**

**Difference Between Registration and Login**

* **Registration**: A user provides new credentials and other necessary information to create an account.
* **Login**: A registered user enters their credentials to access their account.

**2. Valid Login Scenarios**

A valid login scenario ensures that users can successfully log in with the correct credentials.

**Example Scenarios:**

| **Scenario** | **Description** |
| --- | --- |
| **Valid Email & Password** | User logs in using a registered email and correct password. |
| **Valid Phone Number & Password** | User logs in using a registered phone number and correct password. |
| **Phone Number Format Variations** | Different valid formats include country codes (e.g., +20, 0020, or 0123456789). |
| **Remember Me Functionality** | System remembers the login session if the "Remember Me" option is checked. |
| **Multi-Factor Authentication (MFA)** | A user is prompted to enter a verification code after entering the correct credentials. |

**3. Invalid Login Scenarios**

Invalid login scenarios test how the system handles incorrect login attempts.

**Example Scenarios:**

| **Scenario** | **Description** |
| --- | --- |
| **Empty Email/Phone Number Field** | User attempts login without entering an email or phone number. |
| **Empty Password Field** | User attempts login without entering a password. |
| **Incorrect Email/Phone Number** | User enters an email or phone number that is not registered. |
| **Incorrect Password** | User enters the wrong password for a registered account. |
| **Multiple Incorrect Password Attempts** | After multiple failed login attempts, the account is temporarily locked. |
| **Using an Old Password** | Attempting to log in with a previously used password that has been changed. |
| **Invalid Email Format** | User enters an improperly formatted email (e.g., "user@@gmail.com"). |
| **Auto-fill & Password Copying** | System should prevent auto-filling of passwords and copying/pasting sensitive data. |

**4. Additional Login Test Cases**

**Correcting Spelling Mistakes in Emails**

If a user mistypes their email domain (e.g., "gamil.com" instead of "gmail.com"), the system should suggest a correction.

**Password Visibility Toggle**

Users should be able to toggle password visibility by clicking an "eye" icon.

**Login on Different Platforms**

* **Web Application Login**
* **Mobile Application Login**

**Login Security Measures**

1. **Brute Force Protection** - Account lock after multiple failed attempts.
2. **CAPTCHA Implementation** - Preventing bots from attempting logins.
3. **Session Expiry** - Logging out users after inactivity.
4. **Two-Factor Authentication (2FA)** - Enhancing security with additional verification.

**5. Conclusion**

Understanding login test scenarios is essential for ensuring secure authentication. Proper testing guarantees a seamless user experience and robust security against unauthorized access.

This document provides a fundamental understanding of login functionalities and their testing scenarios, making it easier for beginners to grasp the concepts.

**Study Material: Understanding Search Functionalities and Writing Test Scenarios**

**Introduction to Search Functionalities**

Search functionality is a critical component of many applications. It allows users to find relevant information efficiently. This study material explains search functionality, its importance, and how to write test scenarios for it.

**What is Search Functionality?**

Search functionality enables users to retrieve specific information from a database or system by entering keywords or phrases. It is commonly found in:

* E-commerce websites (e.g., searching for a product)
* Learning platforms (e.g., searching for courses or instructors)
* Social networks (e.g., searching for friends or posts)

**Example:**

When you type "laptop" in an e-commerce website’s search bar, the system fetches all products related to laptops based on relevance.

**How Search Works?**

Search functionalities follow a systematic process:

1. **User Input:** The user enters a query in the search bar.
2. **Processing:** The system interprets the input and retrieves relevant results.
3. **Ranking:** The results are sorted based on predefined criteria (e.g., relevance, popularity, price).
4. **Output:** The results are displayed to the user.

**Diagram Representation:**

[User Input] → [Processing] → [Ranking] → [Output]

**Types of Search Functionalities**

| **Type** | **Description** | **Example** |
| --- | --- | --- |
| Keyword Search | Matches exact words in the database | Searching "Python Course" shows courses with "Python" in the title. |
| Fuzzy Search | Identifies similar terms or misspelled words | Searching "Pythn" still shows results for "Python". |
| Filtered Search | Uses categories to refine search results | Searching "Shoes" with a filter for "Men's Shoes". |
| Autocomplete Search | Suggests results while typing | Typing "Lap" suggests "Laptop", "Laptop Bag". |
| Voice Search | Uses voice input to search | Saying "Find Python courses" on a learning platform. |

**Test Scenarios for Search Functionality**

To ensure search functionality works correctly, testers write test scenarios.

**Scenario 1: Searching for a Valid Item**

* **Test Case:** Search for an existing course on Udemy, such as "Data Science".
* **Expected Result:** The system should display all courses related to Data Science.

**Scenario 2: Searching for a Non-Existent Item**

* **Test Case:** Search for a random, non-existing course, like "Alien Language".
* **Expected Result:** The system should display "No results found".

**Scenario 3: Partial Search Queries**

* **Test Case:** Search using part of a course name, like "Develop" for "Web Development".
* **Expected Result:** The system should show all courses containing "Develop".

**Scenario 4: Search with Misspellings**

* **Test Case:** Enter "Data Sience" instead of "Data Science".
* **Expected Result:** The system should suggest "Did you mean Data Science?".

**Scenario 5: Search Using Filters**

* **Test Case:** Search for "Python" and apply the "Beginner" filter.
* **Expected Result:** Only beginner-level Python courses should be displayed.

**Scenario 6: Search in Different Languages**

* **Test Case:** Search for "Software Testing" in Spanish (“Prueba de Software”).
* **Expected Result:** The system should display courses in Spanish if supported.

**Scenario 7: Search Performance Test**

* **Test Case:** Conduct searches for various terms and measure response time.
* **Expected Result:** The system should return results within 2-3 seconds.

**Scenario 8: Voice Search Functionality**

* **Test Case:** Use voice command "Find Python courses".
* **Expected Result:** The system should process the voice command and display Python courses.

**Conclusion**

Search functionality is an integral part of applications, enhancing user experience. Proper testing ensures that users can efficiently find the information they need. By writing detailed test scenarios, testers can ensure the accuracy, relevance, and efficiency of search features.

**Study Material: Test Case Writing (Beginner Level)**

**What is a Test Case?**

A test case is a set of conditions or steps used to verify if a software application functions correctly. Each test case has specific inputs, execution steps, and expected outcomes.

**Why Are Test Cases Important?**

* Ensure software works as expected.
* Identify bugs before deployment.
* Provide a structured way to test applications.
* Facilitate efficient communication between developers and testers.

**Basic Components of a Test Case**

A test case typically includes the following fields:

| **Field** | **Description** |
| --- | --- |
| **Test Case ID** | A unique identifier for the test case (e.g., TC001). |
| **Test Case Title** | A brief description of the test. |
| **Test Description** | Detailed explanation of what the test will verify. |
| **Preconditions** | Any requirements that must be met before executing the test. |
| **Test Steps** | A step-by-step procedure to execute the test. |
| **Test Data** | Input data required for testing. |
| **Expected Result** | The expected output if the application is working correctly. |
| **Actual Result** | The actual output observed after execution. |
| **Status** | Indicates if the test case passed or failed. |

**Example of a Test Case**

**Test Case: Login Functionality**

| **Field** | **Value** |
| --- | --- |
| **Test Case ID** | TC001 |
| **Test Case Title** | Verify login with valid credentials |
| **Test Description** | Ensure the user can log in successfully with correct credentials. |
| **Preconditions** | User must be registered with a valid username and password. |
| **Test Steps** | 1. Open the login page 2. Enter valid username and password 3. Click the login button |
| **Test Data** | Username: testuser, Password: Test@123 |
| **Expected Result** | User is redirected to the dashboard page. |
| **Actual Result** | (To be filled during execution) |
| **Status** | (Pass/Fail) |

**Types of Test Cases**

1. **Functional Test Cases** – Verify if features work as expected.
2. **Negative Test Cases** – Test with invalid inputs to check error handling.
3. **Boundary Test Cases** – Check limits of input fields.
4. **Performance Test Cases** – Validate application speed and responsiveness.
5. **Security Test Cases** – Ensure data protection and secure access.

**Best Practices for Writing Test Cases**

* Use **clear and concise** language.
* Keep test steps **simple and sequential**.
* Use **consistent naming conventions**.
* Ensure each test case is **independent**.
* **Avoid assumptions**, specify all necessary details.
* Write both **positive and negative** test cases.

**Example Diagram: Test Case Process Flow**

Start → Define Test Case → Identify Preconditions → Execute Steps → Compare Expected & Actual Results → Mark Status (Pass/Fail) → End

**Conclusion**

Test case writing is essential for delivering bug-free software. By following structured test cases, testers can efficiently validate software functionality and improve software quality.

**Next Steps**

* Practice writing test cases for different scenarios.
* Learn about automated testing tools like Selenium and JUnit.
* Explore test case management tools like TestRail and Jira.

Happy Testing! 🚀

**Study Material: Writing Test Cases in Google Sheets**

**Introduction to Writing Test Cases**

Testing is an essential part of software development. It ensures that the software works correctly and meets user requirements. Writing test cases is the first step in the testing process.

In this guide, we will learn how to write test cases using **Google Sheets**, a simple and collaborative tool for managing test cases efficiently.

**Why Use Google Sheets for Writing Test Cases?**

Google Sheets is a great tool for test management because:

* It is free and easy to use.
* It allows multiple users to collaborate in real-time.
* It requires no setup or installation.
* It can be customized for different projects and teams.

**Creating a Test Case Document in Google Sheets**

**Step 1: Setting Up the Document**

1. Open **Google Sheets**.
2. Create a new sheet and rename it (e.g., "Test Cases").
3. Create a header section to store general project information:

| **Project Name** | **Designed By** | **Platform** | **Version** | **Delivery Date** |
| --- | --- | --- | --- | --- |
| Sample App | Your Name | Android/iOS/Web | 1.0 | 1 Oct 2020 |

**Step 2: Structuring the Test Cases**

Each test case should have specific columns:

| **Test Case No.** | **Test Suite** | **Test Case Description** | **Steps** | **Expected Result** | **Status** | **Assigned To** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Login | User logs in with valid credentials | 1. Open app2. Enter credentials3. Click login | User is redirected to dashboard | Ready to Test | John Doe |
| 2 | Sign Up | User creates a new account | 1. Open app2. Click sign up3. Enter details & submit | Account is created successfully | Ready to Test | Jane Smith |

**Explanation of Columns:**

* **Test Case No.**: A unique identifier for each test case.
* **Test Suite**: A group of related test cases (e.g., Login, Sign-Up, Checkout).
* **Test Case Description**: A brief explanation of what is being tested.
* **Steps**: The actions needed to perform the test.
* **Expected Result**: The expected outcome after executing the test.
* **Status**: The current state of the test case (e.g., Ready to Test, Pass, Fail, Blocked).
* **Assigned To**: The tester responsible for executing the test.

**Managing Test Cases in Google Sheets**

**Adding More Test Suites**

Each test suite corresponds to a functionality in the application. To create a new test suite:

1. Right-click on an existing sheet (e.g., "Sign Up").
2. Click **Duplicate**.
3. Rename it to the new test suite (e.g., "Cart").

**Color-Coding Status for Better Visibility**

To enhance readability, apply conditional formatting to the **Status** column:

* **Green**: Pass
* **Red**: Fail
* **Yellow**: Blocked
* **Blue**: Ready to Test

Steps:

1. Select the **Status** column.
2. Click **Format > Conditional Formatting**.
3. Set color rules based on text values.

**Collaborating with Your Team**

**Sharing the Test Case Document**

1. Click the **Share** button (top-right corner).
2. Enter team members’ email addresses.
3. Set access permissions:
   * **Editor**: Can modify test cases.
   * **Commenter**: Can add comments.
   * **Viewer**: Can only view the document.
4. Alternatively, create a shareable **link** with appropriate permissions.

**Best Practices for Writing Test Cases**

* **Use Clear and Simple Language**: Test cases should be easy to understand.
* **Be Specific**: Each step should be well-defined.
* **Ensure Coverage**: Cover all major functionalities of the application.
* **Use Unique Identifiers**: Assign unique numbers to test cases for easy reference.
* **Regularly Update the Document**: Keep test cases up to date based on changes in the application.

**Conclusion**

Using Google Sheets for test case management is a simple and effective method for teams. It allows easy collaboration, real-time updates, and structured documentation of test cases. By following this guide, you can efficiently create, manage, and execute test cases for your projects.

**Study Material: Basic Software Testing Concepts and Test Case Writing**

**1. Introduction to Software Testing**

Software testing is the process of evaluating and verifying that a software application or system meets specified requirements. It aims to identify bugs, errors, or missing requirements to ensure quality before deployment.

**Why is Software Testing Important?**

* Ensures the software functions as intended.
* Identifies and fixes bugs before the product reaches users.
* Improves security and reliability.
* Enhances user satisfaction.

**2. Understanding Test Cases**

A test case is a set of conditions, actions, and expected outcomes designed to verify a particular feature or functionality of an application.

**Components of a Test Case**

| **Component** | **Description** |
| --- | --- |
| **Test Case ID** | A unique identifier for the test case. |
| **Test Scenario** | A high-level description of what is being tested. |
| **Preconditions** | Conditions that must be met before the test begins. |
| **Test Steps** | Specific actions taken to execute the test. |
| **Expected Result** | The expected outcome of executing the test steps. |
| **Actual Result** | The observed outcome after execution. |
| **Status** | Pass/Fail/Blocked/Skipped based on the outcome. |

**3. Example: Sign-Up Test Case**

**Scenario: Testing the Sign-Up Functionality**

Sign-up is a critical functionality in any application as it allows users to register and access the system.

**Test Case: Valid Admin Registration**

| **Step No.** | **Test Step** | **Expected Result** |
| --- | --- | --- |
| 1 | Open the sign-up page | Sign-up page is displayed |
| 2 | Click on the "Admin" radio button | "Admin" is selected, "Project Owner" is unselected |
| 3 | Enter a valid admin username | Username is entered successfully |
| 4 | Enter a valid email | Email is entered successfully |
| 5 | Enter a valid password | Password is entered successfully |
| 6 | Click on "Sign Up" | User is redirected to home page, verification email is sent |

**4. Types of Test Cases**

**Positive Test Cases**

* Test scenarios where valid inputs and expected behaviors are checked.
* Example: Entering correct credentials and verifying successful login.

**Negative Test Cases**

* Test scenarios with invalid inputs or unexpected behaviors.
* Example: Entering an invalid email format and verifying the system displays an error message.

**5. Understanding Radio Buttons in UI Testing**

A **radio button** is a type of input field that allows users to select only one option from a set.

**Example:**

* **Admin [○]**
* **Project Owner [○]**
* When the user selects "Admin," "Project Owner" is automatically deselected.

**Diagram:**

[○] Admin [●] Project Owner

* The user can only select one at a time.

**6. Expected Results and Test Status**

* **Pass**: The test case execution matches the expected result.
* **Fail**: The test case execution does not match the expected result.
* **Blocked**: The test case cannot proceed due to dependency on another test or issue.
* **Skipped**: The test was not executed due to certain reasons.

**Example:**

| **Test Case ID** | **Test Scenario** | **Status** |
| --- | --- | --- |
| TC\_001 | Valid Admin Registration | Pass |
| TC\_002 | Invalid Email Format | Fail |
| TC\_003 | Sign-Up Page Not Loading | Blocked |

**7. Conclusion**

Understanding software testing basics and writing effective test cases ensures better software quality. The sign-up functionality is a crucial part of any system, and ensuring its smooth operation enhances user experience and trust.

This study material provides a structured approach to learning software testing with a focus on practical test case writing.

**Basic Study Material on Test Cases and Execution**

**What is a Test Case?**

A test case is a set of conditions or steps used to determine whether a software application functions correctly. It typically includes:

* **Test Case ID** – A unique identifier
* **Test Scenario** – Description of what is being tested
* **Preconditions** – Conditions that must be met before execution
* **Test Steps** – Sequential actions to execute the test
* **Expected Result** – The anticipated outcome
* **Actual Result** – The outcome after execution
* **Status** – Pass, Fail, Blocked, or Skipped

**Example of a Simple Test Case**

| **Test Case ID** | **Test Scenario** | **Steps** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC001 | User login with valid credentials | 1. Open login page 2. Enter valid username and password 3. Click login | User should successfully log in | User successfully logged in | Pass |

**Understanding Test Execution Status**

**1. Pass**

If the actual result matches the expected result, the test case is marked as "Pass."

**2. Fail**

If the actual result differs from the expected result, it is marked as "Fail," and the details can be added to a bug report.

**3. Blocked**

If a test case cannot proceed due to a dependency issue, it is marked as "Blocked."

**4. Skipped**

If a test case is not executed for any reason, it is marked as "Skipped."

**Test Case Execution Flow**

1. **Design Step** – Writing test cases before execution
2. **Execution Step** – Running the test cases and documenting results

**Example Test Cases for User Registration**

**1. Register Using Valid Admin Credentials**

* **Step 1:** Open the registration page
* **Step 2:** Select "Admin" role
* **Step 3:** Enter a valid username and email
* **Step 4:** Click "Register"
* **Expected Result:** The admin account is successfully created

**2. Register Using Valid Project Owner Credentials**

* Similar to the admin case but selecting "Project Owner" instead

**Test Case Maintenance**

Rather than writing new test cases from scratch, existing test cases are modified to suit a new project or application.

**Registration Using Facebook**

**1. Facebook User with an Email**

* **Step 1:** Click on the "Sign in with Facebook" button
* **Step 2:** Login to Facebook
* **Expected Result:** User is registered successfully and redirected to the application

**2. Facebook User with a Phone Number**

* **Step 1:** Click on the "Sign in with Facebook" button
* **Step 2:** Login to Facebook
* **Step 3:** The application prompts for an email
* **Expected Result:** User must enter an email to complete registration

**Importance of Reviewing Test Cases**

Before executing test cases, they should be reviewed to:

* Identify mistakes
* Ensure all scenarios are covered
* Improve clarity and efficiency

**Conclusion**

Test cases are essential for validating software functionality. Proper test execution and maintenance help ensure software reliability and quality. Reviewing test cases before execution prevents common errors and enhances efficiency.

**Basic Study Material on Test Cases and Execution**

**1. What is a Test Case?**

A **test case** is a set of actions, conditions, and expected results that help verify whether a software application is working correctly. It is used to check if a particular function or feature of an application behaves as expected.

**Example of a Test Case:**

| **Test Case ID** | **Test Scenario** | **Steps to Execute** | **Expected Result** |
| --- | --- | --- | --- |
| TC\_001 | User Login | 1. Open login page2. Enter valid email & password3. Click "Login" button | User is successfully logged in |
| TC\_002 | Invalid Login | 1. Open login page2. Enter invalid email & password3. Click "Login" button | Error message "Invalid credentials" appears |

**2. Test Case Components**

A test case generally includes:

1. **Test Case ID** – Unique identifier for each test case.
2. **Test Scenario** – A description of what the test will verify.
3. **Test Steps** – The actions to be performed during the test.
4. **Expected Result** – The anticipated output of the test.
5. **Actual Result** – The actual outcome after execution.
6. **Status** – Whether the test passed or failed.

**3. Test Execution Process**

**Step-by-Step Process**

1. **Requirement Analysis** – Understanding what needs to be tested.
2. **Test Case Creation** – Writing detailed test cases.
3. **Test Case Execution** – Performing the tests on the application.
4. **Defect Reporting** – Reporting any errors or issues found.
5. **Re-testing & Regression Testing** – Rechecking fixed issues and ensuring new changes haven't broken existing functionality.
6. **Test Closure** – Completing documentation and summarizing test results.

**Example: Registration Test Case Execution**

1. Open the registration page.
2. Enter a **valid email** (e.g., [user@example.com](mailto:user@example.com)).
3. Enter a **valid password** (e.g., Password@123).
4. Click **Sign Up**.
5. Expected Outcome: User is registered and redirected to the home page.

**4. Types of Test Cases**

| **Type** | **Description** | **Example** |
| --- | --- | --- |
| **Positive Test Cases** | Tests valid inputs to check if the system works correctly. | Enter valid username & password → Login successfully. |
| **Negative Test Cases** | Tests invalid inputs to check if the system handles errors properly. | Enter incorrect password → Show "Invalid Credentials" error. |
| **Boundary Test Cases** | Tests extreme values to check system behavior. | Enter password with exactly 8 characters. |
| **Usability Test Cases** | Tests ease of use and user experience. | Check if "Sign Up" button is easily visible. |

**5. Registration Test Cases**

**Scenario 1: Register with a Valid Email and Password**

| **Step No.** | **Action** | **Expected Result** |
| --- | --- | --- |
| 1 | Open registration page | Page loads successfully |
| 2 | Enter a valid email | Email is accepted |
| 3 | Enter a valid password | Password is accepted |
| 4 | Click Sign Up | User is redirected to home page |

**Scenario 2: Register Using Google**

| **Step No.** | **Action** | **Expected Result** |
| --- | --- | --- |
| 1 | Click "Sign Up with Google" | Google login page opens |
| 2 | Select a Google account | User is redirected back |
| 3 | Confirm registration | User is successfully signed up |

**Scenario 3: Register with an Already Used Email**

| **Step No.** | **Action** | **Expected Result** |
| --- | --- | --- |
| 1 | Open registration page | Page loads successfully |
| 2 | Enter an email that is already registered | Error message "Email already exists" appears |

**6. Invalid Scenarios and Error Handling**

**Possible Issues During Sign-Up:**

1. **Entering an invalid email format** → Show error "Invalid email format".
2. **Leaving required fields empty** → Show error "This field is required".
3. **Using an already registered email** → Show error "Email already exists".

**7. Best Practices for Writing Test Cases**

* **Use simple and clear language** so anyone can understand.
* **Include all necessary details** (test steps, expected results, etc.).
* **Use unique test case IDs** for easy tracking.
* **Write both positive and negative test cases** to ensure full coverage.
* **Make test cases reusable** for different scenarios.

**Conclusion**

Test cases are an essential part of software testing. They ensure that applications work as expected and help identify bugs early. Well-written test cases improve efficiency and reduce software failures. By following structured test case execution, we can develop reliable and user-friendly applications.

**Study Material: Understanding Invalid Test Cases**

**What is an Invalid Test Case?**

An **invalid test case** is a test scenario that includes incorrect, missing, or unexpected data to check if the software handles errors properly. This ensures that the system does not accept invalid inputs and provides proper error messages to guide the user.

**Example of an Invalid Test Case: Sign-Up Page**

Let's consider a **Sign-Up Page** that requires the following fields:

1. User Type (Admin or Project Owner)
2. Username
3. Email
4. Password

If a user does not provide valid input for one of these fields, the system should display an error message. Each field should be tested separately to identify specific errors.

**Steps to Create an Invalid Test Case**

To write a test case effectively, we follow these steps:

1. **Identify Input Fields:** Check the fields in the form.
2. **Select One Invalid Field:** Leave one field empty or enter incorrect data.
3. **Fill Other Fields Correctly:** Ensure the remaining fields have valid inputs.
4. **Perform the Action:** Click on the "Sign Up" button.
5. **Observe the System Response:** Verify if an appropriate error message appears.
6. **Record the Expected Result:** The system should highlight the invalid field with an error message.

**Example Test Case: Missing User Type**

| **Step No.** | **Action** | **Expected Result** |
| --- | --- | --- |
| 1 | Leave the "User Type" field empty | The field remains blank |
| 2 | Enter a valid username | Username is accepted |
| 3 | Enter a valid email | Email is accepted |
| 4 | Enter a valid password | Password is accepted |
| 5 | Click on "Sign Up" | Error message appears: "Please choose a user type." |

**Expected Error Message:**

The system should display an error message indicating that the user must select a **User Type** before proceeding.

**Why Test One Invalid Field at a Time?**

If multiple fields contain errors, it becomes difficult to identify which field caused the issue. Testing one invalid field at a time helps in isolating and fixing errors efficiently.

**Example:**

* If the **username, email, and password** are invalid at the same time, the system might crash or show a general error message. This makes debugging difficult.
* By testing one invalid field at a time, we can ensure that each field triggers the correct validation message.

**Different Types of Error Messages**

The error message can appear in different formats, such as:

1. **Text Message:** “Please enter a valid email.”
2. **Pop-up Alert:** A dialog box displaying the error.
3. **Highlighted Field:** The invalid field is marked with a red border.
4. **Inline Message:** The error is displayed under the input field.

**Example Diagram of Error Message Display**

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

| User Type: [ ] \*Please select a type|

| Username: [JohnDoe] |

| Email: [johndoe@example.com] |

| Password: [\*\*\*\*\*\*] |

| [Sign Up] |

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

**Conclusion**

Writing invalid test cases ensures that the system behaves correctly under incorrect input conditions. By following a structured approach, we can:

* Identify weak points in the validation process.
* Ensure users receive proper guidance on fixing errors.
* Prevent software crashes and improve usability.

Testing invalid scenarios is an essential part of **Quality Assurance (QA)** and ensures a better user experience. 🚀

**Study Material: Understanding Invalid Test Cases**

**What is an Invalid Test Case?**

An invalid test case is a test scenario designed to check how a system behaves when given incorrect, unexpected, or out-of-bound inputs. These test cases help in identifying potential bugs and ensuring that the system handles errors gracefully.

**Example:**

If a user registration form requires a username of at least 3 characters, entering "A" as a username is an invalid test case because it doesn't meet the minimum requirement.

**Invalid Test Cases for Username Field**

**1. Entering a Username with Less than the Required Characters**

**Scenario:**

* The system requires a username with at least 3 characters.
* User enters a single character, "A".
* Fills in other required fields correctly.
* Clicks on "Sign Up".

**Expected Outcome:**

* An error message appears: "Username must be at least 3 characters long."

**Diagram:**

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

| Username: A |

| Password: \*\*\*\*\*\*\*\* |

| Confirm Password: \*\*\*\*\*\*|

| [Sign Up] |

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

↓

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

| Error: Username must be at least |

| 3 characters long. |

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

**2. Using an Already Taken Username**

**Scenario:**

* The system does not allow duplicate usernames.
* User enters "JohnDoe" which is already taken.
* Fills in other required fields correctly.
* Clicks on "Sign Up".

**Expected Outcome:**

* An error message appears: "Username is already taken. Please choose another one."

**3. Entering an Extremely Long Username**

**Scenario:**

* The system limits username length to 50 characters.
* User enters a 200-character long username.
* Clicks on "Sign Up".

**Expected Outcome:**

* An error message appears: "Username must not exceed 50 characters."

**Table Representation:**

| **Input Length** | **Expected Outcome** |
| --- | --- |
| 1 character | Error: Too short |
| 3-50 characters | Success |
| 200 characters | Error: Too long |

**4. Leaving Username Field Empty**

**Scenario:**

* User does not enter any username.
* Clicks on "Sign Up".

**Expected Outcome:**

* An error message appears: "Username cannot be empty."

**5. Using Special Characters in Username**

**Scenario:**

* The system does not allow special characters in usernames.
* User enters "John@Doe!".
* Clicks on "Sign Up".

**Expected Outcome:**

* An error message appears: "Username cannot contain special characters."

**6. Registering with a Username in a Different Language**

**Scenario:**

* User enters an Arabic username "طارق".
* Clicks on "Sign Up".

**Expected Outcome:**

* If the system supports multi-language usernames, registration is successful.
* Otherwise, an error message appears: "Only Latin characters are allowed."

**Table Representation:**

| **Language** | **Expected Outcome** |
| --- | --- |
| English (JohnDoe) | Success |
| Arabic (طارق) | Depends on system |
| Japanese (ユー) | Depends on system |

**Conclusion**

Invalid test cases ensure that the system behaves correctly under unexpected inputs. By testing various invalid scenarios, we can identify potential issues and improve the reliability of the application.

**Basic Study Material on Email and Username Validation**

**1. What is a Username?**

A **username** is a unique identifier that represents a user in a system. It does not follow any specific format and can contain letters, numbers, and sometimes special characters.

**Example:**

* **Valid Usernames:** john\_doe, user123, Alice-Wonder
* **Invalid Usernames:** (empty), spaces only, special characters only (!@#)

**2. What is an Email?**

An **email address** is used for communication over the internet. It has a specific format that includes:

1. A local part (username before @)
2. The "@" symbol
3. A domain name (example.com)

**Example:**

* **Valid Email:** [alice@example.com](mailto:alice@example.com)
* **Invalid Email:** alice.example.com (missing @), alice@com (missing domain), [alice@.com](mailto:alice@.com) (invalid domain)

**Email Format:**

abc@xyz.com

**Where:**

* abc → Username
* @ → Separator
* xyz.com → Domain name

**3. Why is Email Validation Important?**

Email validation ensures that users enter a correctly formatted email. It prevents errors and enhances security.

**4. Common Email Test Cases**

| **Test Case** | **Scenario** | **Expected Outcome** |
| --- | --- | --- |
| **1. Valid Email** | User enters a correctly formatted email | Accepted |
| **2. Empty Email** | User leaves email field blank | Show error: "Email cannot be empty" |
| **3. Invalid Format** | User enters abc123 or user@com | Show error: "Invalid email format" |
| **4. Already Used Email** | User tries to register with an existing email | Show error: "Email is already in use" |
| **5. Unverified Email** | User signs up with an email that was registered but not verified | Prompt to verify email |
| **6. Email Linked to Social Account** | User tries to register with an email used for Facebook/Google login | Prompt user to login using the linked social account |

**5. Handling Duplicate Emails**

If a user enters an email that is already registered, the system should:

1. Display an error message: "This email is already registered. Please log in."
2. Provide an option to reset the password.

**Example Message:**

This email is already registered.

[Login] [Forgot Password?]

**6. Email Verification Process**

**Why?** To ensure the email belongs to the user.

**Steps:**

1. User registers with an email.
2. System sends a verification link.
3. User clicks the link to verify.
4. Account is activated.

**Example Verification Message:**

Subject: Verify Your Email

Dear User,

Please click the link below to verify your email:

[Verify Email]

Thank you!

**7. Special Cases: Social Media Registration**

If a user previously registered using Facebook/Google and tries to sign up with the same email manually, the system should:

* Prevent duplicate accounts.
* Ask the user to log in with their social account.
* Merge accounts if applicable.

**Example Scenario:**

| **Case** | **User Action** | **System Response** |
| --- | --- | --- |
| Registered with Facebook | Tries to sign up manually | Show message: "This email is linked to Facebook. Log in with Facebook." |
| Registered with Google | Tries to sign up manually | Show message: "This email is linked to Google. Log in with Google." |

**8. Browser and Platform Testing**

Testing should be done on different platforms and browsers to ensure compatibility.

| **Platform** | **Browsers** |
| --- | --- |
| Mobile (Android, iOS) | Chrome, Safari |
| Desktop (Windows, Mac, Linux) | Chrome, Firefox, Edge |

**9. Conclusion**

Validating emails and usernames helps in maintaining security, avoiding duplicate accounts, and improving the user experience. Proper error handling ensures a smooth registration and login process.

**Study Material: Understanding Password Security**

**1. Introduction to Passwords**

A password is a secret combination of characters used to authenticate a user and grant access to a system, application, or website. Secure passwords help protect personal information from unauthorized access.

**2. Password Restrictions and Importance**

Different applications impose various restrictions on passwords to enhance security. Some common restrictions include:

| **Restriction Type** | **Description** |
| --- | --- |
| Minimum Length | Password must be at least six or more characters long. |
| Alphanumeric | Password must include both letters and numbers. |
| Special Characters | Password must contain at least one special character (e.g., @, #, $). |
| Uppercase Letter | Password must start with a capital letter. |

**Why are these restrictions important?**

* Prevents easy-to-guess passwords like "123456" or "password".
* Makes brute-force attacks harder.
* Protects sensitive user data.

**3. Common Password Scenarios & Errors**

When setting up a password, users may encounter different scenarios based on the imposed restrictions.

**3.1. Empty Password Field**

* If a user leaves the password field empty, an error message appears: **"Password cannot be empty."**

**3.2. Password Without Minimum Length**

* If a password is shorter than required (e.g., "Ab12" when the requirement is 6+ characters), an error appears: **"Password must be at least six characters long."**

**3.3. Password Without Numbers**

* Example: "Secure!"
* Error: **"Password must contain at least one number."**

**3.4. Password Without Special Characters**

* Example: "Secure123"
* Error: **"Password must include at least one special character."**

**3.5. Password Without Uppercase Letter**

* Example: "secure!123"
* Error: **"Password must start with a capital letter."**

**4. Best Practices for Secure Passwords**

To ensure strong password security, follow these best practices:

* Use a mix of **uppercase and lowercase** letters.
* Include **numbers** and **special characters**.
* Avoid using **common words** or **personal information** (e.g., "John123").
* Never reuse old passwords.
* Consider using a **password manager** to store and manage complex passwords.

**5. Password Confirmation (Usability Suggestion)**

It is recommended to use two fields for password entry:

1. **Password**
2. **Confirm Password**

**Why?**

* Helps users avoid typing mistakes.
* Reduces the risk of incorrect password entry at login.

**6. Security Considerations**

A weak password can be easily compromised through methods like brute-force attacks, dictionary attacks, or phishing. Implementing strong password policies and two-factor authentication (2FA) can significantly improve security.

**Example of a Strong Password:**

A$trongP@55w0rd!

**7. Test Cases for Password Validation**

| **Test Case No.** | **Scenario** |
| --- | --- |
| TC-01 | Sign up with an empty password. |
| TC-02 | Sign up with a password less than six characters. |
| TC-03 | Sign up with a password missing a number. |
| TC-04 | Sign up with a password missing a special character. |
| TC-05 | Sign up with a password that does not start with a capital letter. |

These test cases ensure that password validation works as expected and maintains security standards.

**8. Conclusion**

Passwords are a critical part of digital security. By implementing strong password policies and educating users on best practices, applications can significantly reduce the risk of unauthorized access and data breaches.