**Submitted By:**

TAYYABA AFZAL (2022-CS-134)

RAVEEHA MOHSIN (2022-CS-149)

Testing Phase (SDLC)

Table of Contents

[2 Overview: 3](#_Toc182210335)

[3 Software Test Techniques 3](#_Toc182210336)

[1. Black Box Testing 3](#_Toc182210337)

[2. White Box Testing 3](#_Toc182210338)

[3. Grey Box Testing 4](#_Toc182210339)

[4 Types of Software Testing 4](#_Toc182210340)

[4.1 Functional Testing 4](#_Toc182210341)

[4.2 Non-Functional Testing 4](#_Toc182210342)

[5 Categories of Functional Testing 4](#_Toc182210343)

[Unit Testing 5](#_Toc182210344)

[5.1.1 Test Scenarios 5](#_Toc182210345)

[5.2 Integration Testing 10](#_Toc182210346)

[5.2.1 Test Scenarios 10](#_Toc182210347)

[System Testing 14](#_Toc182210348)

[5.2.2 Test scenarios 14](#_Toc182210349)

[Acceptance Testing 17](#_Toc182210350)

[5.2.3 Test Scenarios 18](#_Toc182210351)

[6 Categories of Non-Functional Testing 21](#_Toc182210352)

[6.1 Performance Testing 21](#_Toc182210353)

[6.1.1 Test Scenarios 21](#_Toc182210354)

[6.2 Usability Testing 22](#_Toc182210355)

[6.2.1 Test Scenarios 22](#_Toc182210356)

[6.3 Security Testing 24](#_Toc182210357)

[6.3.1 Test Scenarios 24](#_Toc182210358)

[6.4 Compatibility Testing 25](#_Toc182210359)

[6.4.1 Test Scenarios 26](#_Toc182210360)

[6.5 Reliability Testing 27](#_Toc182210361)

[6.5.1 Test Scenarios 27](#_Toc182210362)

[6.6 System Design Documentation 28](#_Toc182210363)

[6.6.1 Test Scenarios 29](#_Toc182210364)

[7 Types of Testing Based on Effort Involved 36](#_Toc182210365)

[7.1 Manual Testing 36](#_Toc182210366)

[7.1.1 Test scenarios 36](#_Toc182210367)

[7.2 Automated Testing 37](#_Toc182210368)

[7.2.1 Test scenarios 38](#_Toc182210369)

[8 Other Types of Software Testing 39](#_Toc182210370)

[8.1 Alpha and Beta Testing 39](#_Toc182210371)

[8.1.1 Test Scenarios 40](#_Toc182210372)

[8.2 Exploratory Testing 42](#_Toc182210373)

[8.2.1 Testing Scenarios 42](#_Toc182210374)

[8.3 Regression Testing 43](#_Toc182210375)

[8.3.1 Test scenarios 43](#_Toc182210376)

[8.4 End-to-End Testing 45](#_Toc182210377)

[8.4.1 Testing Scenarios 45](#_Toc182210378)

[9 Conclusion 46](#_Toc182210379)

Testing Phase

# Overview:

The testing phase is a critical stage in the software development lifecycle where the developed software is evaluated to ensure it functions as intended, meets specified requirements, and is free from major bugs or issues. This phase involves running various tests to identify and fix any defects, validate the system’s performance, and ensure it meets the user’s expectations.

For the "**Career Path Navigator**" project, the testing phase would focus on verifying that the AI recommendations for career paths are accurate and relevant to the user’s skills and interests. It would also involve testing the functionality of features like the skills and interests input, career suggestions, skill gap analysis, and the counselor interaction system. The testing will ensure that the backend (SQL Server) is reliably storing and retrieving user information, and that the system remains responsive, secure, and user-friendly on all platforms.

# Software Test Techniques

Software test techniques are approaches used to evaluate the performance, functionality, and reliability of a software system. These techniques help identify defects and ensure that the software meets quality standards before deployment. The three most common types of software test techniques are as follows:

1. Black Box Testing  
   Black box testing is a technique that assesses a software system’s functionality without considering its internal code structure. Testers focus on the inputs provided to the system and the expected outputs, simulating the user’s perspective. This technique is used to validate the system’s external behavior against the specified requirements.
2. White Box Testing  
   White box testing, also known as structural or code-based testing, examines the software’s internal operations and code structure. This approach allows developers to verify the system’s logic, performance, and efficiency from a technical standpoint, ensuring that each component functions as intended.
3. Grey Box Testing  
   Grey box testing combines elements of both black box and white box testing. Testers have partial knowledge of the system’s internal components, allowing them to evaluate specific aspects of the system’s logic and functionality without a full code analysis. This technique helps uncover issues that may be missed by purely external or internal testing methods.

# Types of Software Testing

Software testing can be broadly categorized into two main types: **Functional Testing** and **Non-Functional Testing**. These types aim to ensure that the software operates according to the specified requirements and meets both technical and user expectations.

## Functional Testing

Functional testing is focused on validating the software system's functionality, ensuring that it performs its intended operations as per the defined technical specifications. This type of testing assesses whether the software meets the functional requirements set by stakeholders.

## Non-Functional Testing

Non-functional testing evaluates the non-functional aspects of the software system. It focuses on testing how the system performs its functions, rather than what the system does. Key areas assessed include performance, usability, security, scalability, and reliability.

# Categories of Functional Testing

Functional testing can be further divided into several key categories, each focusing on different aspects of the software’s functionality. These categories include:

Unit Testing  
Unit testing examines individual components or modules of the software to ensure they work correctly in isolation. Typically, unit testing is performed by developers during the coding phase to verify that each unit of code functions as expected.

### Test Scenarios

#### Test Scenario 01: User Registration Functionality

* **Objective**: Verify that users can successfully register with valid inputs.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** Unit Testing
* **Steps**:
  + Enter valid name, email, and password.
  + Click on the "Register" button.
* **Expected Result**: User receives a confirmation email, and registration completes successfully.
* **Pass Criteria**:
  + Required fields are validated.
  + Email format is checked.
  + Password requirements (e.g., minimum length, special characters) are enforced.

#### Test Scenario 02: Duplicate Email Registration

* **Objective**: Ensure the system prevents registration with an already-registered email.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** Unit Testing
* **Steps**:
  + Register with an email already in the system.
  + Attempt to submit the form.
* **Expected Result**: Error message displays, and registration is denied.
* **Pass Criteria**:
  + Duplicate email check.
  + Clear error messaging for duplicate emails.

#### Password Strength Validation

* **Objective**: Validate the strength of the password during registration.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** System Testing
* **Steps**:
  + Attempt registration with weak passwords (e.g., less than the minimum length, no special characters).
* **Expected Result**: The system denies registration with a weak password.
* **Pass Criteria**:
  + Password complexity (length, special characters, uppercase/lowercase letters).
  + User-friendly feedback for weak password.

#### Test Scenario 03: Login with Valid Credentials

* **Objective**: Verify login functionality with correct credentials.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** Unit Testing
* **Steps**:
  + Enter registered email and correct password.
  + Click "Login."
* **Expected Result**: User is redirected to the dashboard or homepage.
* **Pass Criteria**:
  + Successful login redirection.
  + Session is established upon successful login.

#### Test Scenario 04: Login with Incorrect Password

* **Objective**: Ensure the system denies login for incorrect passwords.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** Unit Testing
* **Steps**:
  + Enter registered email and incorrect password.
  + Click "Login."
* **Expected Result**: Error message displays, and login is denied.
* **Pass Criteria**:
  + Error message for incorrect password.
  + Lockout after multiple failed attempts (if applicable).

#### Test Scenario 04: Password Reset Functionality

* **Objective**: Verify the functionality of password reset.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** Unit Testing
* **Steps**:
  + Request password reset with a valid email.
  + User will go to their respective dashboard after login and then click “update” to rest password.
* **Expected Result**: User successfully resets their password and can log in with the new password.
* **Pass Criteria**:
  + Email is sent for password reset.
  + Link in email redirects correctly and allows secure reset.

#### Test Scenario 06: Profile Information Completion

* **Objective**: Ensure students can fill out their personal and background information accurately.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** Unit Testing
* **Steps**:
  + Navigate to the Profile section.
  + Enter personal and educational background details.
  + Save the information.
* **Expected Result**: Information is saved successfully and displayed upon returning to the profile.
* **Pass Criteria**:
  + Required fields are validated.
  + Data persistence upon saving.
  + Display of saved information for review.

#### Test Scenario 07: Validate Error Handling for Missing Mandatory Fields in Profile Creation

* **Objective:** Ensure that the system prompts for mandatory fields if left blank.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** Unit Testing
  + **Input:** Leave required fields blank and attempt to save the profile.
  + **Expected Result:** System displays an error message for each missing required field.
  + **Steps:**
    1. Attempt to save the profile with missing required fields.
  + **Pass Criteria:** Error messages should appear, specifying the missing required fields.

#### Test Scenario 08: Validate Scheduling a Meeting with a Student

* **Objective:** Test the counselor’s ability to schedule a meeting with a student.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** Unit Testing
  + **Input:** Counselor selects a student, date, and time for the meeting.
  + **Expected Result:** The meeting is scheduled successfully, and confirmation is displayed.

**Steps:**

* + Navigate to the “Schedule Meeting” section.
  + Select a student from the list.
  + Choose a date and time slot from the calendar interface.
  + Confirm the meeting details by clicking “Schedule.”
  + Review the confirmation message for scheduled meeting details.

**Pass Criteria:**

* + A confirmation message displays the meeting details (student name, date, time).
  + The scheduled meeting should appear under both the counselor’s and student’s “Upcoming Meetings.”
  + Notifications should be sent to the student and counselor regarding the scheduled meeting.

## **Integration Testing**

Integration testing focuses on evaluating the interactions between two or more integrated components of the system. The objective is to identify issues related to data flow, communication, and the interface between integrated components.

### Test Scenarios

#### Test Scenario 01: Validate Ability to Send Notifications to Students and Counselors

* **Objective:** Test the admin’s functionality to send notifications to users (students and counselors).
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** Integration Testing
  + **Input:** Admin drafts a notification and selects recipients.
  + **Expected Result:** Notifications are sent successfully, and a confirmation message is displayed.

**Steps:**

* + Open the “Notifications” section in the admin dashboard.
  + Compose a notification message.
  + Choose recipients (specific users, groups, or all users).
  + Click “Send” and confirm the action.
  + Verify notification delivery by checking sample student/counselor accounts.

**Pass Criteria:**

* + A confirmation message displays the number of recipients.
  + Notifications appear in the recipient accounts promptly.
  + The notification message content is accurately displayed with no truncation or format issues.

#### Test Scenario 02: Editing or Cancelling a Meeting

* **Objective**: Allow students to modify or cancel scheduled meetings.
* **Testing Type:** Functional Testing
* **Functional Testing Type:** CRUD Operations Testing (Create, Read, Update, Delete)
* **Steps**:
  + Navigate to Scheduled Meetings.
  + Select a meeting and choose to edit or cancel.
  + Confirm action.
* **Expected Result**: Meeting is updated or removed, with notifications sent as appropriate.
* **Pass Criteria**:
  + Data persistence of edits or cancellation.
  + Notifications of changes to involved parties.
  + Prevention of last-minute cancellations (if implemented).

#### Test Scenario 03: Validate Removing a Counselor from the System

* **Objective:** Test the admin’s ability to remove an existing counselor from the system.
* **Testing Type:** Grey Box Testing
* **Functional Testing Type:** Integration Testing
  + **Input:** Admin selects a counselor and confirms deletion.
  + **Expected Result:** The counselor is removed, and a confirmation message is displayed.

**Steps:**

* + Navigate to the “Counselor List” section.
  + Select the counselor to remove and click the “Delete” or “Remove” button.
  + Confirm the removal when prompted.
  + Review the confirmation message indicating the counselor was removed successfully.

**Pass Criteria:**

* + The confirmation message displays the counselor’s name and successful removal.
  + The counselor’s profile should no longer appear in the list.
  + Any unsuccessful attempts to remove should provide an error message with troubleshooting steps.

#### Test Scenario 04: Career Recommendations Based on Interests

* **Objective**: Ensure recommendations are tailored based on selected interests.
* **Testing Type:** Grey Box Testing
* **Functional Testing Type:** Integration Testing
* **Steps**:
  + Complete the Interests section.
  + Check the career recommendation section.
* **Expected Result**: Displayed career recommendations align with the user's interests.
* **Pass Criteria**:
  + Correctness of recommendations.
  + User’s ability to browse and interact with recommendations.

#### Test Scenario 05: Validate Viewing Student Profile Information

* **Objective:** Ensure that counselors can view student profiles and assess personal details, background, and interests.
* **Testing Type:** Grey Box Testing
* **Functional Testing Type:** Integration Testing
  + **Input:** Click on a specific student’s profile from a list.
  + **Expected Result:** The student’s profile information (personal details, background, and interests) is displayed correctly.

**Steps:**

* + From the dashboard, navigate to the “Student Profiles” section.
  + Click on a student’s profile to view their details.
  + Check that all sections of the profile (personal details, background, interests) are displayed accurately.

**Pass Criteria:**

* + All student details should load without delay.
  + The displayed data should match the information the student provided.
  + Any data retrieval issues should trigger an error message, and the counselor should be able to retry loading the profile.

#### Test Scenario 06: Database Storage for Student Data

* **Objective**: Ensure student details, preferences, and progress data are accurately stored in the database.
* **Testing Type:** Functional Testing
* **Functional Testing Type:** Data Integrity Testing
* **Steps**:
  + Complete profile, select interests, and save progress.
* **Expected Result**: Data is accurately saved and retrievable on subsequent logins.
* **Pass Criteria**:
  + Data consistency across sessions.
  + Prevention of data loss or corruption.
  + Speed and efficiency of data retrieval.

System Testing  
System testing involves testing the entire integrated system, including both software and hardware components, to verify that it meets the specified requirements. This testing is typically performed in an environment that simulates real-world conditions.

### Test scenarios

#### Test Scenario 01: Validate Admin Login and Access to Admin Dashboard

* **Objective:** Test the functionality of admin login and access to the admin dashboard.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** System Testing
  + **Input:** Admin enters valid username and password.
  + **Expected Result:** Admin logs in successfully and is redirected to the admin dashboard.

**Steps:**

* + Open the login page and enter the admin’s valid username and password.
  + Click on the “Login” button.
  + Verify that the system authenticates the admin and redirects them to the admin dashboard.

**Pass Criteria:**

* + The admin should gain access to the dashboard after successful login.
  + All dashboard features should load without errors.
  + Incorrect credentials should trigger an appropriate error message without allowing access.

#### Test Scenario 02: Interest Selection

* **Objective**: Validate the selection of career interests for personalized recommendations.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** System Testing
* **Steps**:
  + Open the Interests section.
  + Select multiple interests.
  + Save the selected interests.
* **Expected Result**: Interests are saved, and recommendations align with chosen interests.
* **Pass Criteria**:
  + Selection limits, if any.
  + Data persistence and display accuracy.
  + Impact on career recommendations.

#### Test Scenario 03: Validate Counselor Login and Access to Dashboard

* **Objective:** Test the functionality of counselor login and access to the counselor dashboard.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** System Testing
  + **Input:** Counselor enters valid username and password.
  + **Expected Result:** Counselor logs in successfully and is redirected to the dashboard.

**Steps:**

* + Open the login page and enter the counselor’s valid username and password.
  + Click the “Login” button.
  + Verify that the system authenticates the counselor and redirects them to their dashboard.

**Pass Criteria:**

* + The counselor should be able to access the dashboard upon successful login.
  + All dashboard features should load correctly without errors.
  + Any invalid credentials should trigger an appropriate error message without granting access.

#### Test Scenario 04: Validate Ability to Add New Counselor

* **Objective:** Test the admin’s ability to add a new counselor to the system.
* **Testing Type:** White Box Testing
* **Functional Testing Type:** Integration Testing
  + **Input:** Admin enters counselor details and submits the form.
  + **Expected Result:** The new counselor is successfully added, and confirmation is displayed.

**Steps:**

* + Navigate to the “Add New Counselor” section.
  + Fill in required details (e.g., name, email, qualifications).
  + Click the “Submit” button to add the counselor.
  + Check for a confirmation message indicating successful addition.

**Pass Criteria:**

* + A confirmation message displays the new counselor’s name.
  + The newly added counselor should appear in the “Counselor List” with accurate details.
  + Any missing or incorrect inputs should prompt error messages, guiding the admin to complete the form correctly.

#### Test Scenario 05: Validate Access to System Reports and Analytics

* **Objective:** Test that the admin can access and view reports and analytics on the system’s usage, counselor performance, and student engagement.
* **Testing Type:** Black Box Testing
* **Functional Testing Type:** System Testing
  + **Input:** Click on “Reports” and select specific data for analysis.
  + **Expected Result:** The system loads reports and analytics as selected.

**Steps:**

* + Navigate to the “Reports & Analytics” section in the admin dashboard.
  + Select specific report parameters (e.g., date range, counselor, student engagement metrics).
  + Click “Generate” or “View” to display the reports.
  + Review the report content for accuracy and clarity.

**Pass Criteria:**

* + The selected report is generated and displayed within a reasonable loading time.
  + Data in the report accurately reflects system usage metrics.
  + No data errors or missing fields appear in the report; the report should update promptly as per input changes.

Acceptance Testing  
Acceptance testing is performed by the client or end-user to ensure that the system meets business requirements and is ready for deployment. This type of testing is often based on real business scenarios and is performed before the software is officially accepted for production.

### Test Scenarios

#### Test Scenario 01: User Authentication for Students

* **Objective:** Verify that students can successfully authenticate and access the system using their credentials.
* **Testing Type:** Acceptance Testing
* **Functional Testing Type:** User Authentication Testing
* **Steps:**
  + Navigate to the login page.
  + Enter valid student credentials (username and password).
  + Click on the login button.
* **Expected Result:** The student is authenticated and directed to their dashboard.
* **Pass Criteria:**
  + Successful login with valid credentials.
  + Error message displayed for invalid credentials.
  + Session maintained until logout.

#### Test Scenario 02: Career Path Recommendations Based on Interests

* **Objective:** Ensure that the system provides relevant career path recommendations based on student inputs (interests, background, etc.).
* **Testing Type:** Acceptance Testing
* **Functional Testing Type:** Recommendation System Testing
* **Steps:**
  + Log in as a student.
  + Provide personal details, background, and interests.
  + Click on "Get Career Recommendations."
* **Expected Result:** System displays career recommendations based on the student’s profile.
* **Pass Criteria:**
  + Accurate recommendations aligned with user inputs.
  + Recommendations are varied and realistic.
  + No errors during the recommendation process.

#### Test Scenario 03: Feedback Submission for the System

* **Objective:** Verify that students can successfully submit feedback for the system’s usability and functionality.
* **Testing Type:** Acceptance Testing
* **Functional Testing Type:** Feedback Submission Testing.
* **Steps:**
  + Log in as a student.
  + Navigate to the "Feedback" section.
  + Provide a review about the system.
  + Submit the feedback.
* **Expected Result:** Feedback is successfully submitted, and a confirmation message is displayed.
* **Pass Criteria:**
  + Feedback is submitted without errors.
  + Confirmation message is displayed.
  + Feedback is saved for future retrieval.

#### Test Scenario 04: Meeting Scheduling with a Counsellor

* **Objective:** Ensure students can schedule meetings with a counsellor based on their availability.
* **Testing Type:** Acceptance Testing
* **Functional Testing Type:** Meeting Scheduling Testing
* **Steps:**
  + Log in as a student.
  + Navigate to the "Counsellor Meeting" section.
  + Choose a date and time.
  + Confirm the meeting schedule.
* **Expected Result:** The meeting is successfully scheduled, and both the student and counsellor receive a confirmation.
* **Pass Criteria:**
  + Meeting time is correctly scheduled.
  + No conflicts in meeting times.
  + Notifications are sent to both student and counsellor.

#### Test Scenario 05: Viewing and Editing Student Profile

* **Objective:** Validate that students can view and edit their personal profile information.
* **Testing Type:** Acceptance Testing
* **Functional Testing Type:** Profile Management Testing
* **Steps:**
  + Log in as a student.
  + Navigate to the "Profile" section.
  + View personal details such as name, email, and interests.
  + Edit details (e.g., update interests) and save changes.
* **Expected Result:** Profile is updated, and changes are visible after saving.
* **Pass Criteria:**
  + Personal information is displayed correctly.
  + Changes are saved and updated in the database.
  + Profile update does not cause any errors.

# Categories of Non-Functional Testing

Non-Functional testing can be further divided into several key categories, each focusing on different aspects of the software’s functionality. These categories include:

## Performance Testing

In performance testing, the efficacy of an application will be examined by applying pressure. They will only be concerned with performance characteristics such as responsiveness, capacity, flexibility, and stability.

### Test Scenarios

#### Test Scenario 01: Validate System Response Time Under Load

* **Objective:** Ensure the application responds within an acceptable time when accessed by multiple users simultaneously.
* **Testing Type:** Performance Testing (Load Testing)
  + **Input:** Simulate 500 concurrent users accessing the system.
  + **Expected Result:** The application should respond within 2 seconds for each request.

**Steps:**

* + Use a performance testing tool (e.g., JMeter) to simulate 500 users logging in and navigating the dashboard.
  + Monitor the response time for each request and note any delays.
  + Record response times and analyze for consistency.

**Pass Criteria:**

* + Response time should be ≤ 2 seconds under load.
  + The system should not crash or show degraded performance with 500 concurrent users.

#### Test Scenario 02: Validate Data Retrieval Time for Reports

* **Objective:** Test the time taken to retrieve and display large datasets (such as user analytics reports).
* **Testing Type:** Performance Testing (Stress Testing)
  + **Input:** Generate and load a report with data spanning a 1-year period.
  + **Expected Result:** Report should be displayed within 5 seconds.

**Steps:**

* + Access the “Reports & Analytics” section.
  + Select a report with a large dataset (e.g., a full year of user data).
  + Measure the time from when the report is requested to when it is displayed.

**Pass Criteria:**

* + The report should load in ≤ 5 seconds.
  + The system should handle high data loads without crashes or timeouts.

## Usability Testing

Usability testing analyzes an application’s user-friendliness and identifies flaws in the system’s end-user framework.

### Test Scenarios

#### Test Scenario 01: Validate Ease of Navigation

* **Objective:** Ensure that users can easily navigate between different sections of the system.
* **Testing Type:** Usability Testing
  + **Input:** Task a user to locate and perform various functions, such as accessing student profiles or scheduling a meeting.
  + **Expected Result:** User should complete tasks without confusion or difficulty.

**Steps:**

* + Ask a user to log in and locate the “Student Profiles” section.
  + Direct the user to schedule a meeting and view analytics.
  + Observe the user’s interactions and note any difficulties in navigation.

**Pass Criteria:**

* + Users should be able to find sections intuitively without additional guidance.
  + All navigational elements should be clearly labeled and accessible.

#### Test Scenario 02: Validate Mobile and Web Usability

* **Objective:** Ensure that the interface is intuitive and accessible on both desktop and mobile devices.
* **Testing Type:** Usability Testing, Compatibility Testing
  + **Input:** Access the system on various devices (desktop, tablet, mobile) and perform common tasks.
  + **Expected Result:** The interface should be easy to use and visually coherent across all devices.

**Steps:**

* + Access the application on desktop, tablet, and mobile browsers.
  + Complete tasks such as logging in, viewing profiles, and scheduling meetings.
  + Check that all elements are correctly displayed, and navigation is consistent.

**Pass Criteria:**

* + The interface should adapt responsively to different screen sizes.
  + All core functionalities should be usable on both desktop and mobile.

#### Test Scenario 03: Chatbot Interaction for Career Queries

* **Objective**: Verify chatbot functionality for providing guidance.
* **Testing Type:** Functional Testing
* **Functional Testing Type:** Usability Testing / AI Testing
* **Steps**:
  + Open the chatbot interface.
  + Enter a query regarding career paths.
* **Expected Result**: Chatbot responds with relevant information.
* **Pass Criteria**:
  + Prompt response time.
  + Accuracy and relevance of response.
  + Flow and continuity of conversation.

## Security Testing

Security testing is a form of software testing which evaluates a software application’s security. It assists in identifying system vulnerabilities and ensures that confidential data is safeguarded.

### Test Scenarios

#### Test Scenario 01: Validate User Authentication and Authorization

* **Objective:** Ensure only authorized users can access sensitive sections of the system.
* **Testing Type:** Security Testing
  + **Input:** Attempt unauthorized access to restricted areas (e.g., Admin accessing Counselor’s dashboard).
  + **Expected Result:** Unauthorized users should be restricted and receive an appropriate error message.

**Steps:**

* + Log in as a user with restricted permissions (e.g., student).
  + Attempt to access admin and counselor-only features.
  + Check that the system restricts access and displays an error message.

**Pass Criteria:**

* + Unauthorized users should receive an error message and be blocked from accessing restricted areas.
  + All roles should only access sections designated for their permissions.

#### Test Scenario 02: Validate Data Encryption for Sensitive Information

* **Objective:** Ensure that sensitive data, such as passwords and personal details, are encrypted.
* **Testing Type:** Security Testing
  + **Input:** Check the storage and transmission methods for sensitive data.
  + **Expected Result:** All sensitive data should be encrypted in the database and during transmission.

**Steps:**

* + Inspect the database and identify if sensitive fields (passwords, contact information) are encrypted.
  + Use a network monitoring tool to check if sensitive data is encrypted during transmission.

**Pass Criteria:**

* + Passwords and sensitive information should be stored and transmitted in encrypted formats.
  + The system should use HTTPS for secure communication.

## Compatibility Testing

The purpose of compatibility testing is to examine the app’s functionality in a particular software and hardware ecosystem. Only after the app is functionally stable should compatibility testing commence.

### Test Scenarios

#### Test Scenario 01: Validate Cross-Browser Compatibility

* **Objective:** Ensure the application performs consistently across multiple browsers.
* **Testing Type:** Compatibility Testing
  + **Input:** Access the application using various browsers (Chrome, Firefox, Edge, Safari).
  + **Expected Result:** The application should perform identically across all browsers.

**Steps:**

* + Open the application on Chrome, Firefox, Edge, and Safari.
  + Test basic functionalities, including login, navigation, and data entry.
  + Compare the user experience across browsers.

**Pass Criteria:**

* + All functionalities should work consistently across all supported browsers.
  + There should be no visual discrepancies or functionality limitations across different browsers.

#### Test Scenario 02: Validate Compatibility with Different Screen Resolutions

* **Objective:** Ensure that the application interface adapts to different screen resolutions.
* **Testing Type:** Compatibility Testing, Usability Testing
  + **Input:** Access the application on devices with various screen resolutions (e.g., 1024x768, 1920x1080).
  + **Expected Result:** The application layout should adapt to different resolutions without distortion.

**Steps:**

* + Open the application on screens with different resolutions.
  + Navigate through the application to check layout consistency.
  + Verify that no functionality or text is distorted.

**Pass Criteria:**

* + The layout should display correctly across different resolutions.
  + No overlapping or cut-off elements should appear on any screen resolution.

## Reliability Testing

Reliability Testing is a type of software testing that focuses on ensuring a software system performs consistently and without failure under expected conditions over a specified period. The goal is to validate that the software is dependable, stable, and provides consistent performance in real-world scenarios. Reliability testing identifies potential weaknesses or areas where the software might fail, ensuring that any potential issues are addressed before deployment.

### Test Scenarios

#### Test Scenario 01: Validate System Recovery After Unexpected Shutdown

* **Objective:** Ensure the application can recover without data loss after an unexpected shutdown.
* **Testing Type:** Reliability Testing
  + **Input:** Simulate a sudden system shutdown or crash during operation.
  + **Expected Result:** The system should restart and recover data accurately.

**Steps:**

* + Log into the system and perform actions like data entry or scheduling.
  + Simulate an unexpected shutdown.
  + Restart the application and verify data consistency.

**Pass Criteria:**

* + System should recover to its last known state without data loss.
  + All user data should remain consistent post-recovery.

#### Test Scenario 02: Validate System Stability Under Continuous Use

* **Objective:** Ensure system stability and performance after prolonged usage.
* **Testing Type:** Reliability Testing (Endurance Testing)
  + **Input:** Continuously use the system over a 24-hour period with multiple users.
  + **Expected Result:** The application should remain stable without crashing.

**Steps:**

* + Simulate a 24-hour test with multiple users logging in, accessing data, and performing tasks.
  + Monitor for any memory leaks, system lags, or crashes.
  + Record system performance and stability over time.

**Pass Criteria:**

* + The system should remain responsive and free of crashes.
  + No memory leaks or performance degradation should occur.

## System Design Documentation

System Design Documentation Testing is a quality assurance process that involves reviewing and validating the design documentation for a software system to ensure it meets specified requirements and accurately represents the intended architecture, functionality, and interactions of the system components. This type of testing helps verify that the design documentation is comprehensive, clear, and free from ambiguities, errors, or inconsistencies, which ultimately supports accurate and effective system development.

### Test Scenarios

#### Test Scenario 01: Validate Completeness of Design Documentation

* **Objective:** Ensure that the design documentation includes all necessary diagrams and descriptions for each component of the system.
* **Testing Type:** Documentation Review
  + **Input:** Review the design documentation for completeness.
  + **Expected Result:** Documentation should include all required diagrams (Use Case, Class, Sequence, Activity, etc.) and detailed descriptions.

**Steps:**

* + Check the table of contents to verify the inclusion of all design diagrams.
  + Review each section to ensure all diagrams are present and correctly labeled.
  + Verify that each diagram has a descriptive heading and accompanying explanation.

**Pass Criteria:**

* + All required diagrams and descriptions should be present and accurately labeled.
  + Each diagram should have an explanation of its purpose and functionality in the system.

#### Test Scenario 02: Validate Accuracy of Diagrams

* **Objective:** Ensure that all diagrams accurately represent the system's design and functionality.
* **Testing Type:** Design Verification
  + **Input:** Compare the diagrams with project requirements and functionality.
  + **Expected Result:** Diagrams should correctly depict system components, interactions, and data flows.

**Steps:**

* + Review Use Case, Sequence, Class, and Activity Diagrams against project requirements.
  + Verify that each component, function, and interaction in the system is correctly represented.
  + Ensure there are no missing or incorrect elements in each diagram.

**Pass Criteria:**

* + Diagrams should accurately reflect all system requirements and interactions.
  + No discrepancies should exist between the diagrams and the specified functionality.

#### Test Scenario 03: Validate Consistency Across Diagrams

* **Objective:** Ensure consistency in terms, symbols, and components across all design diagrams.
* **Testing Type:** Consistency Check
  + **Input:** Review terminology and symbols used across all diagrams.
  + **Expected Result:** Terms and symbols should be used consistently across all diagrams.

**Steps:**

* + Check for consistent naming conventions for classes, objects, and attributes.
  + Verify that the same symbols (e.g., arrows for interactions) are used across all diagrams.
  + Ensure any relationships or dependencies are depicted consistently.

**Pass Criteria:**

* + All diagrams should use consistent terminology, symbols, and visual representations.
  + No conflicting terms or symbols should appear between diagrams.

Test Scenario 04: Validate Coverage of Functional Requirements in Use **Case Diagram**

* **Objective:** Ensure all functional requirements are represented in the Use Case Diagram.
* **Testing Type:** Requirements Validation
  + **Input:** Compare the use cases in the diagram with documented functional requirements.
  + **Expected Result:** Each functional requirement should correspond to at least one use case.

**Steps:**

* + Review the functional requirements list.
  + Compare each requirement with the use cases in the diagram.
  + Check if any requirement is missing from the use case representation.

**Pass Criteria:**

* + All functional requirements should have corresponding use cases in the diagram.
  + No functional requirement should be left out.

#### Test Scenario 05: Validate Actor-Use Case Interactions

* **Objective:** Ensure that all actors and their interactions with use cases are accurately represented.
* **Testing Type:** Interaction Verification
  + **Input:** Check each actor’s interactions with related use cases.
  + **Expected Result:** All actors should have appropriate connections to relevant use cases.

**Steps:**

* + Identify each actor in the system (e.g., Student, Counselor, Admin).
  + Review their interactions with use cases in the diagram.
  + Ensure that each interaction correctly represents how the actor interacts with the system.

**Pass Criteria:**

* + Each actor should have correct associations with relevant use cases.
  + No incorrect or missing actor-use case relationships should be present.

#### Test Scenario 06: Validate Class Structure and Attributes

* **Objective:** Ensure each class in the system has the correct structure, attributes, and relationships.
* **Testing Type:** Structural Verification
  + **Input:** Review the Class Diagram and compare it with the system’s data model.
  + **Expected Result:** All classes should have correct attributes, methods, and relationships.

**Steps:**

* + Verify that each class has all required attributes and methods.
  + Check that relationships between classes (e.g., inheritance, aggregation) are correct.
  + Ensure that data types and visibility (public, private) are accurate.

**Pass Criteria:**

* + Each class should have the appropriate attributes, methods, and relationships.
  + No attribute or method should be missing or incorrect.

#### Test Scenario 07: Validate Consistency of Class Diagram with ERD

* **Objective:** Ensure that the Class Diagram aligns with the Entity-Relationship Diagram (ERD).
* **Testing Type:** Consistency Check
  + **Input:** Compare Class Diagram with ERD attributes and relationships.
  + **Expected Result:** Both diagrams should represent the same entities and relationships.

**Steps:**

* + Compare entities in the ERD with classes in the Class Diagram.
  + Verify that relationships (e.g., one-to-many) match in both diagrams.
  + Check that any attributes listed in the ERD appear in the Class Diagram.

**Pass Criteria:**

* + Classes and relationships should match entities and relationships in the ERD.
  + No inconsistencies between the two diagrams.

#### Test Scenario 08: Validate Sequence of Operations for Key Use Cases

* **Objective:** Ensure that the Sequence Diagrams accurately represent the order of operations for key use cases.
* **Testing Type:** Flow Verification
  + **Input:** Compare Sequence Diagrams with use case flows and requirements.
  + **Expected Result:** Sequence Diagrams should accurately depict the flow of operations.

**Steps:**

* + Select key use cases (e.g., User Login, Counselor Meeting Scheduling).
  + Review the Sequence Diagram to verify the order of operations.
  + Ensure that each step aligns with the corresponding use case flow.

**Pass Criteria:**

* + Sequence of operations should accurately match the use case flow.
  + All interactions should be correctly represented with no missing steps.

#### Test Scenario 09: Validate Messaging Between Components

* **Objective:** Ensure that messages between objects/components are accurately represented.
* **Testing Type:** Interaction Verification
  + **Input:** Check the Sequence Diagram for all messages exchanged between components.
  + **Expected Result:** Each message should be labeled accurately with the correct operation.

**Steps:**

* + Identify all interactions between components in the Sequence Diagram.
  + Verify that each message has the correct operation name.
  + Check for missing or incorrect messages.

**Pass Criteria:**

* + All messages should accurately reflect operations between components.
  + No messages should be missing or incorrectly labeled.

#### Test Scenario 10: Validate Workflow for Key Processes

* **Objective:** Ensure that Activity Diagrams represent the correct workflows for key processes.
* **Testing Type:** Workflow Verification
  + **Input:** Compare the Activity Diagram with the documented process flows.
  + **Expected Result:** Each activity should align with the expected process flow.

**Steps:**

* + Select key processes (e.g., Registration, Career Recommendation).
  + Compare the documented flow with the Activity Diagram.
  + Ensure that each activity in the flow is represented accurately.

**Pass Criteria:**

* + Activity Diagrams should represent the correct process flows without deviations.
  + Each workflow should cover all required steps and decision points.

#### Test Scenario 11: Validate Decision Points in Activity Diagrams

* **Objective:** Ensure that all decision points (e.g., conditions) are accurately represented in Activity Diagrams.
* **Testing Type:** Decision Verification
  + **Input:** Review decision points in Activity Diagrams for logical correctness.
  + **Expected Result:** Each decision should have clear conditions leading to correct outcomes.

**Steps:**

* + Identify all decision points in Activity Diagrams.
  + Verify that each decision has clearly defined conditions.
  + Check that outcomes from each decision point are logically consistent.

**Pass Criteria:**

* + Decision points should have well-defined conditions with correct outcomes.
  + No ambiguity in decision logic or outcomes.

# Types of Testing Based on Effort Involved

Software testing can also be classified based on the level of effort required. The two main types in this category are **Manual Testing** and **Automated Testing**. These types differ in how the tests are executed, with manual testing requiring human involvement and automated testing relying on pre-written scripts and tools.

## Manual Testing

Manual testing involves a tester manually navigating through the application, interacting with the software code, or using approved testing tools to evaluate the system's behavior. This approach requires human judgment and is commonly used for exploratory testing, user interface testing, and when automation is not feasible or cost-effective. Manual testing ensures that the application meets the requirements and identifies issues from an end-user's perspective.

### Test scenarios

#### Test Scenario 01: Manual Testing for Career Recommendation Accuracy

* **Objective**: Verify that the career recommendations generated for students align with their input skills, interests, and the latest industry trends.
* **Testing Type**: Manual Functional Testing
* **Steps**:
  + Navigate to the Career Recommendation section.
  + Input a variety of skills and interests (e.g., “data analysis,” “machine learning,” “web development”).
  + Submit the form and wait for recommendations to be generated.
  + Review the recommended career paths and ensure they are logically aligned with the input.
* **Expected Result**: The recommendations should accurately reflect the student’s skill set and interests.
* **Pass Criteria**:
  + The career paths suggested should be relevant to the input.
  + No irrelevant career options should appear.

#### Test Scenario 02: Manual Testing for Feedback Submission

* **Objective**: Verify the functionality of the feedback form and ensure that submitted feedback is correctly stored.
* **Testing Type**: Manual Functional Testing
* **Steps**:
  + Navigate to the Feedback section.
  + Fill in the feedback form with appropriate comments and ratings.
  + Submit the feedback form.
  + Check whether the feedback is stored correctly and a confirmation message is displayed.
* **Expected Result**: The feedback should be stored in the system, and a confirmation message should appear.
* **Pass Criteria**:
  + The system should validate the input fields (e.g., ratings and text fields) before submission.
  + The feedback should appear in the database or backend system, ready for review.

## Automated Testing

Automated testing utilizes tools and scripts to automatically run predefined test cases. This testing approach is typically employed in continuous integration and continuous delivery (CI/CD) environments, where frequent and repetitive tests need to be executed efficiently. Automated testing helps improve testing speed, accuracy, and coverage by running a large number of tests quickly and without human intervention. It is particularly beneficial for regression testing and performance testing.

### Test scenarios

#### Test Scenario 01: Automated Testing for Login Functionality

* **Objective**: Ensure that the login functionality works as expected for valid and invalid credentials.
* **Testing Type**: Automated Functional Testing
* **Steps**:
  + Automate login using valid and invalid usernames and passwords.
  + Run the test script with various combinations of valid and invalid credentials.
  + Check for appropriate success and error messages.
* **Expected Result**:
  + A successful login should redirect to the user dashboard.
  + Invalid login attempts should display an error message ("Invalid username or password").
* **Pass Criteria**:
  + For valid credentials, the user should be able to log in successfully.
  + For invalid credentials, an error message should be displayed.

#### Test Scenario 02: Automated Testing for Meeting Scheduler Functionality

* **Objective**: Ensure that the meeting scheduler correctly schedules, updates, and cancels meetings.
* **Testing Type**: Automated Functional Testing (CRUD Operations)
* **Steps**:
  + Automate the process of scheduling a meeting by selecting a date, time, and counselor.
  + Test updating the meeting (e.g., changing time or date).
  + Test cancellation of the meeting.
  + Ensure that notifications are sent for scheduling, updates, and cancellations.
* **Expected Result**:
  + The meeting is successfully scheduled with the selected counselor.
  + Updates to meeting details are applied without issues.
  + A cancellation removes the meeting and sends a notification to both the student and counselor.
* **Pass Criteria**:
  + Meetings should be scheduled, updated, and canceled correctly.
  + Notifications should be triggered for each operation (schedule, update, cancel).

# Other Types of Software Testing

In addition to the primary types of software testing, there are several other notable testing methods that are critical for ensuring comprehensive quality assurance and a well-functioning software system. These types of testing focus on specific aspects of the software development lifecycle and can help uncover unique issues.

## Alpha and Beta Testing

* **Alpha Testing**  
  Alpha testing is an internal testing phase conducted by the software development team within the organization. The goal is to identify as many bugs and issues as possible before releasing the software to external testers. It often involves a limited group of users who test the software in a controlled environment.
* **Beta Testing**  
  Beta testing occurs after alpha testing and involves releasing the software to a selected group of clients or external users. These testers provide feedback based on real-world usage, helping to identify issues that might not have been discovered during alpha testing. Beta testing ensures that the product is ready for broader public release.

### Test Scenarios

#### Alpha Testing Scenario 01: Career Recommendation System (Alpha)

* **Objective**: Ensure the career recommendation system works in the development environment before being released to external users.
* **Testing Type**: Alpha Testing (Internal Testing)
* **Steps**:
  + Test career recommendations with a variety of input data (different skills, experience levels, and interests).
  + Verify that the system provides relevant career suggestions for each input set.
  + Simulate edge cases (e.g., very specific or unusual skill sets).
* **Expected Result**: Career recommendations should be accurate, based on the input.
* **Pass Criteria**:
  + The career paths generated should align with the entered skills and industry trends.

#### Alpha Testing Scenario 02: AI Chatbot Responses (Alpha)

* **Objective**: Validate that the AI chatbot interacts with users effectively and provides appropriate responses in a controlled environment.
* **Testing Type**: Alpha Testing (Internal Testing)
* **Steps**:
  + Interact with the AI chatbot using common user queries.
  + Test a variety of questions related to career paths, available resources, and general inquiries.
  + Check whether the chatbot provides useful, accurate, and understandable responses.
* **Expected Result**: The AI chatbot should provide accurate responses and guide users appropriately.
* **Pass Criteria**:
  + The chatbot should handle all expected user queries with relevant answers.

#### Beta Testing Scenario 01: User Interface (UI) Usability (Beta)

* **Objective**: Get feedback from real users on the user interface of the Career Path Navigator tool.
* **Testing Type**: Beta Testing (External User Testing)
* **Steps**:
  + Distribute the product to a group of users (students and career counselors).
  + Ask users to navigate the UI, focusing on key features like profile setup and career recommendation.
  + Collect feedback on design, ease of use, and any difficulties encountered.
* **Expected Result**: Users should be able to navigate the UI without significant issues.
* **Pass Criteria**:
  + Positive feedback on UI usability and suggestions for minor improvements.

#### Beta Testing Scenario 02: Feedback Collection (Beta)

* **Objective**: Ensure that the feedback system works as expected with external users.
* **Testing Type**: Beta Testing (External User Testing)
* **Steps**:
  + Provide the beta version of the system to users, allowing them to submit feedback after using the tool.
  + Ensure that users can submit feedback easily, and it is correctly stored in the backend.
  + Test the feedback interface for responsiveness, clarity, and completeness.
* **Expected Result**: Feedback should be submitted without issues, and users should feel comfortable using the system.
* **Pass Criteria**:
  + Feedback should be correctly stored and processed.

## Exploratory Testing

Exploratory testing is an informal and unscripted testing technique where testers actively explore the application using their domain knowledge and experience. Testers do not follow predefined test cases but instead focus on learning about the system and identifying any potential issues. Test guides or reference sheets are often used to assist the tester in focusing on key areas, but the process remains highly flexible.

### Testing Scenarios

#### Testing Scenario 01: User Profile Creation

* **Objective**: Discover potential issues when creating a user profile and inputting different combinations of data.
* **Testing Type**: Exploratory Testing
* **Steps**:
  + Test creating user profiles with different combinations of skills, interests, and experience levels.
  + Use invalid data, such as special characters or empty fields, to see how the system handles it.
  + Navigate through the profile creation flow without predefined test cases.
* **Expected Result**: The system should handle all inputs gracefully, including edge cases and invalid data.
* **Pass Criteria**:
  + The profile creation process should not break or behave unpredictably.

#### Testing Scenario 02: Career Recommendation Flow

* **Objective**: Manually explore the career recommendation process and identify any unexpected behaviors or bugs.
* **Testing Type**: Exploratory Testing
* **Steps**:
  + Input various combinations of skills and interests into the recommendation system.
  + Check if the system outputs career paths that are relevant and accurate.
  + Test random sequences of interactions, such as skipping steps or providing unexpected inputs.
* **Expected Result**: The system should still recommend valid careers, even with unexpected interactions.
* **Pass Criteria**:
  + No crashes, incorrect recommendations, or unexpected behaviors should occur.

## Regression Testing

Regression testing is performed to ensure that recent changes or updates to the software, such as bug fixes or new features, do not introduce new defects or negatively impact existing functionality. The objective is to verify that the software continues to perform as expected after changes have been made, helping to maintain the software’s stability over time.

### Test scenarios

#### Testing Scenario 01: Profile Editing After Update

* **Objective**: Ensure that changes to the profile do not affect existing functionality or cause regressions in the system.
* **Testing Type**: Regression Testing
* **Steps**:
  + Edit an existing user profile, changing skills and interests.
  + Verify that the career recommendations and other functionalities still work as expected after the edit.
  + Ensure that no features or sections (e.g., feedback, meeting schedule) are broken after profile updates.
* **Expected Result**: All functionalities should continue to work as expected after profile updates.
* **Pass Criteria**:
  + No broken features after profile changes.

#### Testing Scenario 02: Job Recommendations After System Update

* **Objective**: Ensure that recent system updates or bug fixes do not break the job recommendation feature.
* **Testing Type**: Regression Testing
* **Steps**:
  + After the system has been updated or new features have been added, test the job recommendation feature.
  + Verify that job recommendations still align with users' skills and interests.
  + Ensure that no new bugs or issues have appeared in previously working features.
* **Expected Result**: The job recommendation system should work as expected after updates.
* **Pass Criteria**:
  + The system should not introduce new issues or errors after updates.

## End-to-End Testing

End-to-end testing simulates real-world user scenarios to validate the functionality of the entire software system from start to finish. It ensures that all components of the system, including integrated third-party services, work together as expected in a production-like environment. This type of testing is critical to ensure the smooth operation of the system in a real user context.

### Testing Scenarios

#### Testing Scenario 01: User Registration to Career Recommendation

* **Objective**: Validate the full user flow from registration to receiving career recommendations.
* **Testing Type**: End-to-End Testing
* **Steps**:
  + Register a new user in the system (fill in all required fields).
  + Complete the profile with skills and interests.
  + Generate career recommendations based on the completed profile.
  + Verify that the entire process works smoothly from start to finish.
* **Expected Result**: The user should be able to register, complete their profile, and receive appropriate career recommendations.
* **Pass Criteria**:
  + The entire process, from registration to career recommendations, should function without issues.

#### Testing Scenario 02: Feedback Submission and Admin View

* **Objective**: Ensure that a user can submit feedback, and the feedback is accessible and viewable by the admin.
* **Testing Type**: End-to-End Testing
* **Steps**:
  + Submit feedback through the system’s feedback form.
  + Verify that the submitted feedback is correctly saved in the backend.
  + Check that the admin can view the feedback from the admin panel.
* **Expected Result**: Feedback should be submitted correctly and visible to the admin.
* **Pass Criteria**:
  + The feedback system should function from submission to admin review without errors.

#### Test Scenario 03: Progress Tracking

* **Objective**: Verify the ability to track progress towards career goals.
* **Testing Type:** Functional Testing
* **Functional Testing Type:** End-to-End Testing
* **Steps**:
  + Begin with initial recommendations.
  + Complete relevant milestones or tasks.
  + Check progress tracking updates.
* **Expected Result**: Progress tracker updates with each completed milestone.
* **Pass Criteria**:
  + Accurate progress calculation.
  + Visual and textual indicators for progress.
  + Data persistence of completed milestones.

# Conclusion

Software testing is an essential process that compares actual outcomes to expected results to ensure that software systems are free from defects and meet the specified requirements. Understanding and applying the 20 essential types of testing (and more) can significantly improve the software's quality and reliability. By selecting the appropriate testing techniques and following best practices, development teams can identify issues early and enhance the overall performance of the software.