****

Contents

[**1.** **Specification-based testing?** 4](#_Toc153722442)

[**2.** **Key aspects:** 4](#_Toc153722443)

[**2.1.** **Requirement Analysis:** 4](#_Toc153722444)

[**2.2.** **Test Planning:** 4](#_Toc153722445)

[**2.3.** **Test Case Design:** 4](#_Toc153722446)

[**3.** **Importance of Specification-based Testing?** 5](#_Toc153722447)

[**3.1.** **Requirement Validation:** 5](#_Toc153722448)

[**3.2.** **Customer Satisfaction:** 5](#_Toc153722449)

[**3.3.** **Early Detection of Defects:** 5](#_Toc153722450)

[**3.4.** **Comprehensive Test Coverage:** 5](#_Toc153722451)

[**3.5.** **Documentation Compliance:** 5](#_Toc153722452)

[**4.** **Specification-based in SDLC:** 6](#_Toc153722453)

[**5.** **Types of Specification-based and their roles:** 6](#_Toc153722454)

[**5.1.** **Requirements Specifications:** 6](#_Toc153722455)

[**5.2.** **Functional Specifications:** 6](#_Toc153722456)

[**5.3.** **Design Specifications:** 7](#_Toc153722457)

[**5.4.** **User Interface Specifications:** 7](#_Toc153722458)

[**5.5.** **Performance Specifications:** 7](#_Toc153722459)

[**5.6.** **Security Specifications:** 8](#_Toc153722460)

[**5.7.** **Database Specifications:** 8](#_Toc153722461)

[**5.8.** **Interoperability Specifications:** 8](#_Toc153722462)

[**5.9.** **Compliance Specifications:** 9](#_Toc153722463)

[**6.** **Advantages of Using Specification-Based Testing:** 9](#_Toc153722464)

[**6.1.** **Alignment with Requirements:** 9](#_Toc153722465)

[**6.2.** **Early Defect Detection:** 9](#_Toc153722466)

[**6.3.** **Objective Evaluation:** 9](#_Toc153722467)

[**6.4.** **Documentation and Traceability:** 9](#_Toc153722468)

[**6.5.** **Facilitates Regression Testing:** 9](#_Toc153722469)

[**6.6.** **User Expectation Validation:** 10](#_Toc153722470)

[**6.7.** **Regulatory Compliance:** 10](#_Toc153722471)

[**6.8.** **Risk Mitigation:** 10](#_Toc153722472)

[**6.9.** **Cost-Effective Testing:** 10](#_Toc153722473)

[**6.10.** **Quality Assurance:** 10](#_Toc153722474)

[**7.** **Limitations of Specification-Based Testing:** 10](#_Toc153722475)

[**7.1.** **Incomplete or Ambiguous Specifications:** 10](#_Toc153722476)

[**7.2.** **Limited Exploration of Internal Logic:** 10](#_Toc153722477)

[**7.3.** **Dependency on Documentation Quality:** 10](#_Toc153722478)

[**7.4.** **Difficulty in Handling Changes:** 10](#_Toc153722479)

[**7.5.** **Limited Scope for Creativity:** 11](#_Toc153722480)

[**7.6.** **May Miss Usability Issues:** 11](#_Toc153722481)

[**8. Specification-based testing techniques:** 11](#_Toc153722482)

[**8.1. Equivalence Partitioning:** 11](#_Toc153722483)

[**8.2. Boundary Value Analysis (BVA):** 11](#_Toc153722484)

[**8.3. Decision Tables:** 12](#_Toc153722485)

[**9.** **Specification Review:** 12](#_Toc153722486)

[**9.1.** **Login Functionality** 12](#_Toc153722487)

[**9.1.1.** **Overview** 12](#_Toc153722488)

[**9.1.2.** **Key Components** 13](#_Toc153722489)

[**9.1.3.** **Expected Behavior** 13](#_Toc153722490)

[**10.** **Equivalence partitioning:** 13](#_Toc153722491)

[**10.1.** **Equivalent classes** 13](#_Toc153722492)

[**10.2.** **Test Cases:** 14](#_Toc153722493)

[**11.** **Boundary Value Analysis** 15](#_Toc153722494)

[**11.1.** **Boundary Value Analysis for login functionality:** 15](#_Toc153722495)

[**11.2.** **Boundary values for input parameter:** 15](#_Toc153722496)

[**11.3.** **Test cases focusing on boundary values:** 16](#_Toc153722497)

[**12.** **Decision Tables** 18](#_Toc153722498)

[**12.1.** **Decision Table for Transferring Funds:** 18](#_Toc153722499)

[**12.2.** **Conditions:** 19](#_Toc153722500)

[**12.3.** **Actions:** 19](#_Toc153722501)

[**12.4.** **Test cases based on decision table** 19](#_Toc153722502)

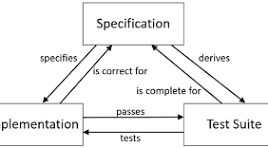
[**13.** **References** 21](#_Toc153722503)

(perforce, 2023)(Rosencrance, n.d.)(geeksforgeeks, geeksforgeeks, n.d.)(javatpoint, n.d.)(Andrew, 2023)

**Q 1: Provide a brief definition of specification-based testing. Explain why specification-based testing is essential in the software testing process.**

# **Specification-based testing?**

Specification-based testing, also known as black-box testing or functional testing, is a testing approach that focuses on verifying whether a software system or component meets its specified requirements. In this method, the tester does not have access to the internal code or implementation details of the software being tested. Instead, they base their test cases on the external specifications, requirements, and functionality outlined in the software's documentation. (Thomas, 2023)



# **Key aspects:**

## **Requirement Analysis:**

Thoroughly understand and analyze the project requirements or specifications.

## **Test Planning:**

Develop a test plan outlining the testing strategy, scope, resources, and schedule based on the specifications.

## **Test Case Design:**

Create test cases that cover different scenarios outlined in the specifications. Include positive and negative test cases.

* 1. **Equivalence Partitioning:**

Divide input and output data into equivalence classes to ensure representative test coverage.

* 1. **Boundary Value Analysis:**

Test at the edges or boundaries of input domains to identify potential errors.

* 1. **Decision Table Testing:**

Create decision tables to systematically test different combinations of input conditions.

* 1. **State Transition Testing:**

For systems with different states, design tests to cover state transitions as specified in the requirements.

* 1. **Use Case Testing:**

Verify the system against use cases described in the specifications.

* 1. **Traceability Matrix:**

Establish a traceability matrix to link test cases back to specific requirements, ensuring comprehensive coverage.

* 1. **Positive and Negative Testing**

Verify that the system behaves correctly under expected conditions and handles unexpected conditions gracefully.

* 1. **Defect Reporting:**

Document and communicate any discrepancies between expected and actual outcomes, facilitating the debugging process. (geeksforgeeks, geeksforgeeks, n.d.)

# **Importance of Specification-based Testing?**

Specification-based testing is essential for several reasons:

## **Requirement Validation:**

It helps ensure that the software meets the specified requirements and behaves according to the documented expectations. This is crucial for validating the accuracy and completeness of the software's functionality. (Thomas, 2023) (altexsoft, 2023)

## **Customer Satisfaction:**

Testing based on specifications helps ensure that the software aligns with customer expectations and requirements. This contributes to customer satisfaction by delivering a product that meets their needs.

## **Early Detection of Defects:**

By systematically testing against documented specifications, defects and issues can be identified early in the development process. This allows for timely resolution and reduces the cost and effort associated with fixing defects in later stages.

## **Comprehensive Test Coverage:**

Specification-based testing facilitates comprehensive test coverage by guiding the testing process based on the features and functionalities outlined in the specifications. This helps ensure that all critical aspects of the software are tested. (javatpoint, n.d.)

## **Documentation Compliance:**

It helps in verifying whether the software conforms to the documented specifications, ensuring that the development team follows the agreed-upon requirements.

**Q2: Discuss different types of specifications used in software development (e.g., functional specifications, design specifications, requirements specifications). Explain how each type of specification contributes to the testing process.**

# **Specification-based in SDLC:**

In software development, various types of specifications are created to document different aspects of the software's design, functionality, and requirements. Each type of specification serves a specific purpose and contributes to the testing process in distinct ways. Here are some common types of specifications and their roles in the testing process (altexsoft, 2023) (lambdatest, 2023)

# **Types of Specification-based and their roles:**

## **Requirements Specifications:**

Requirements specifications outline the functional and non-functional requirements of the software. They describe what the software is supposed to accomplish from a user's perspective. (lambdatest, 2023)

**Contribution to Testing:**

Requirements specifications serve as the foundation for testing. Testers derive test cases from these specifications to verify that the software meets the specified functional and performance criteria. Testing against requirements helps ensure that the software aligns with the intended purpose and user expectations. (springer, 2020)

## **Functional Specifications:**

Functional specifications provide detailed information about the features and functionalities of the software. They describe how the system will accomplish the tasks outlined in the requirements.

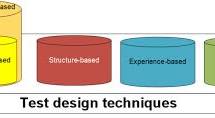


**Contribution to Testing:**

Testers use functional specifications to design test cases that validate each functional aspect of the software. These specifications guide the creation of test scenarios to ensure comprehensive coverage of the system's capabilities. Functional specifications help verify that the software performs as intended and meets user needs.

## **Design Specifications:**

Design specifications describe the internal structure, architecture, and components of the software. They provide details about how the system will be implemented to meet the specified requirements. (Andrew, 2023)



**Contribution to Testing:**

While design specifications are not typically the primary source for test case derivation, they can be useful in certain testing scenarios. Testers may refer to design specifications to gain insights into the system's architecture, identify potential weak points, and design tests that target specific components or interactions. This can be particularly relevant for integration testing and system testing. (Rosencrance, n.d.)

## **User Interface Specifications:**

UI specifications focus on the visual and interactive aspects of the software, detailing how users will interact with the system's user interface. (lambdatest, 2023)

**Contribution to Testing:**

UI specifications guide usability testing and user interface testing. Testers design test cases to verify that the UI elements are consistent with the design, are user-friendly, and provide a positive user experience. UI specifications are crucial for ensuring that the software is not only functional but also user-friendly and intuitive. (Sharma, 2021)

## **Performance Specifications:**

Performance specifications define the expected performance characteristics of the software, such as response times, scalability, and resource utilization.

**Contribution to Testing:**

Performance testing, including load testing, stress testing, and scalability testing, is based on performance specifications. Testers design test scenarios to assess the software's performance against the specified criteria, ensuring that it can handle the expected load and perform efficiently under various conditions.

## **Security Specifications:**

Security specifications outline the security requirements and measures that must be implemented to safeguard the software from potential threats and vulnerabilities. (lambdatest, 2023)

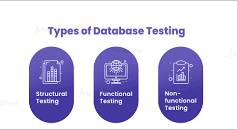


**Contribution to Testing:**

Security testing is based on security specifications. Testers design test cases to assess the software's resistance to unauthorized access, data breaches, and other security risks. This type of testing helps ensure that the software is robust and complies with security standards. (altexsoft, 2023)

## **Database Specifications:**

Database specifications describe the structure, relationships, and operations related to the database used by the software. They include details about data storage, retrieval, and manipulation.



**Contribution to Testing:**

Database testing relies on these specifications to validate the integrity of data storage and retrieval processes. Testers design test cases to ensure that the software interacts correctly with the database, handles data consistency, and performs database operations accurately. (geeksforgeeks, geeksforgeeks, n.d.)

## **Interoperability Specifications:**

Interoperability specifications define the requirements for the software to interact seamlessly with other systems, software, or hardware components.

**Contribution to Testing:**

Interoperability testing is based on these specifications. Testers design test scenarios to verify that the software can exchange data and functionality with external systems or components as required. This type of testing ensures that the software can operate in a heterogeneous environment.

## **Compliance Specifications:**

Compliance specifications outline the standards, regulations, or industry-specific requirements that the software must adhere to. This may include legal, regulatory, or industry standards and guidelines. (lambdatest, 2023) (geeksforgeeks, geeksforgeeks, n.d.)

**Contribution to Testing:**

Compliance testing ensures that the software complies with the specified regulations and standards. Testers design test cases to verify that the software meets legal and industry-specific requirements, helping organizations avoid legal issues and adhere to relevant standards. (lambdatest, 2023)

**Q3. Identify and discuss the benefits of using specification-based testing. Highlight the potential challenges or limitations associated with specification-based testing.**

# **Advantages of Using Specification-Based Testing:**

Efficient validation of requirements, early defect detection, objective evaluation, comprehensive test coverage, and regulatory compliance, promoting reliable software development.

* 1. **Alignment with Requirements:**

Ensures software meets specified requirements, validating functionality against documented expectations. (geeksforgeeks, geeksforgeeks, n.d.) (springer, 2020)

* 1. **Early Defect Detection:**

By testing against specified requirements, defects can be identified early in the development process. This allows for prompt resolution, reducing the cost and effort associated with fixing issues in later stages of development or post-release.

* 1. **Objective Evaluation:**

Focuses on external behavior, providing an impartial assessment of the software without requiring knowledge of internal code.

* 1. **Documentation and Traceability:**

Facilitates the creation of detailed documentation and traceability matrices, aiding in understanding test coverage and supporting maintenance.

* 1. **Facilitates Regression Testing:**

Specification-based testing is conducive to regression testing, where previously tested functionalities are retested to ensure that new changes or updates do not introduce defects. Testers can easily rerun test cases based on the specifications to validate the software's stability after modifications.

* 1. **User Expectation Validation:**

Verifies that the software aligns with user expectations and business needs as outlined in the specifications.

* 1. **Regulatory Compliance:**

Ensures that the software complies with industry regulations and standards specified in the requirements.

* 1. **Risk Mitigation:**

Systematically identifies and mitigates potential risks, contributing to a more proactive risk management approach.

* 1. **Cost-Effective Testing:**

By focusing on specified requirements, SBT optimizes testing efforts, making the testing process more efficient and cost-effective.

* 1. **Quality Assurance:**

Contributes to overall quality assurance by systematically validating software against documented specifications.

# **Limitations of Specification-Based Testing:**

Possible limitations of Specification-Based Testing (SBT) include incomplete coverage, difficulty handling dynamic systems, and reliance on accurate specifications.

* 1. **Incomplete or Ambiguous Specifications:**

Incomplete or ambiguous specifications can lead to incomplete or inaccurate test cases. If the specifications are unclear or open to interpretation, testers may miss critical test scenarios or design test cases that do not accurately reflect the intended behavior. (springer, 2020) (Sharma, 2021)

* 1. **Limited Exploration of Internal Logic:**

Specification-based testing does not involve access to the internal code or implementation details. As a result, it may miss certain defects related to the internal logic of the software. This limitation is addressed by complementary testing approaches such as white-box testing.

* 1. **Dependency on Documentation Quality:**

The effectiveness of specification-based testing heavily relies on the quality of the documentation. Poorly written or outdated specifications can lead to ineffective testing and may result in the acceptance of software with defects.

* 1. **Difficulty in Handling Changes:**

When specifications change, it can be challenging to update test cases accordingly. Managing changes in requirements or specifications may require significant effort, especially if the testing process is already underway.

* 1. **Limited Scope for Creativity:**

Specification-based testing tends to be systematic and may limit the tester's ability to explore creative and unforeseen scenarios. This could potentially lead to overlooking unconventional use cases or real-world scenarios that are not explicitly covered in the specifications.

* 1. **May Miss Usability Issues:**

While specification-based testing ensures that the software functions as intended, it may not capture all usability issues or user experience concerns. Specialized testing, such as usability testing, may be needed to address these aspects.

While specification-based testing offers many advantages in terms of clarity, objectivity, and early defect detection, it is essential to be mindful of its limitations and to supplement it with other testing approaches to ensure a thorough and effective testing process.

**Q4. Describe at least three specification-based testing techniques (e.g., equivalence partitioning, boundary value analysis, decision tables). Provide a brief explanation of each technique and its application.**

# **8. Specification-based testing techniques:**

## **8.1. Equivalence Partitioning:**

Equivalence partitioning involves dividing the input data into groups or partitions and selecting representative test cases from each partition. The idea is that if a system behaves correctly for one set of inputs in a partition, it should behave correctly for all inputs in that partition. Equivalence partitions are typically based on the assumption that elements within a partition are equivalent in terms of expected behavior. (Sharma, 2021)

**Application:**

For example, if a system requires numeric input, equivalence partitioning would involve creating partitions such as valid numeric input, invalid non-numeric input, and boundary values. Test cases are then designed to cover each partition, ensuring that the software handles different types of input appropriately.

## **8.2. Boundary Value Analysis (BVA):**

Boundary Value Analysis focuses on testing the boundaries of input domains. It assumes that defects often occur near the edges or boundaries of valid input ranges. Test cases are designed to include values at the lower and upper boundaries, as well as just below and just above these boundaries. The goal is to identify any issues related to boundary conditions that may lead to incorrect behavior.

**Application:**

For instance, if a system accepts values between 1 and 100, boundary value analysis would involve testing values like 0, 1, 2, 99, 100, and 101. This helps ensure that the software handles the edge cases correctly and does not exhibit unexpected behavior at the boundaries.

## **8.3. Decision Tables:**

Decision tables are a systematic and structured way of representing complex business rules or logical conditions. They provide a matrix that correlates different input conditions with corresponding actions or outcomes. Test cases are derived by considering various combinations of input conditions to cover different scenarios.

**Application:**

Suppose a software system has different discount rules based on the customer type (e.g., regular or premium) and the purchase amount. A decision table would list these conditions and the corresponding discount rates for each combination. Test cases would then be designed to cover various scenarios, ensuring that the software correctly applies discounts based on the specified conditions.

**Q5. Scenario: Consider a simple online banking application. The application allows users to log in, view account balances, transfer funds, and update personal information.**

* 1. **Specification Review**
     1. **Provide a sample functional specification for the login functionality of the online banking application.**
     2. **Identify key components such as input fields, buttons, and expected behavior.**
  2. **Equivalent Partitioning:**
     1. **Apply the equivalence partitioning technique to the login functionality.**
     2. **Identify and list equivalence classes for different input parameters.**
     3. **Design test cases based on the identified equivalence classes.**
  3. **Boundary Value Analysis**
     1. **Perform boundary value analysis for the login functionality.**
     2. **Identify boundary values for input parameters.**
     3. **Design test cases focusing on boundary values.**
  4. **Decision Tables**
     1. **Create a decision table to represent the scenarios for transferring funds in the online banking application.**
     2. **Define relevant conditions and corresponding actions.**
     3. **Generate test cases based on the decision table.**

# **Specification Review:**

## **Login Functionality**

### **Overview**

The login functionality enables users to access their online banking accounts securely. Users are required to provide valid credentials to authenticate and gain access to their accounts.

### **Key Components**

1. **Input fields:**
   * Username/Account number
     + Type: Text
     + Constraints: Required field
     + Validation: Validation will be applied as needed
   * Password
     + Type: Password
     + Constraints: Required field
     + Validation: Validation will be applied as needed
2. **Buttons** 
   * Login
     + Type: Button
     + Action: Initiates the login process upon click
     + Validation: Ensure all required fields are filled correctly.
   * Forgot password
     + Type: Link/Button
     + Action: Redirects to the password recovery/reset page.

### **Expected Behavior**

* Upon loading the login page, the focus should be on the username/account number input field.
* Users should not be able to submit the form unless both the username/account number and password fields are filled.
* After clicking the "Login" button, the system should validate the entered credentials.
* If the credentials are valid, the user should be redirected to their account dashboard.
* If the credentials are invalid, display an error message indicating that the login attempt failed.
* Implement account lockout mechanism after a specified number of unsuccessful logins attempts to enhance security.
* Provide a "Forgot Password" link/button that redirects users to a page for password recovery/reset.

# **Equivalence partitioning:**

## **Equivalent classes**

* 1. Username
     1. Valid Username/Account Number
     2. Invalid Username/Account Number (empty)
     3. Invalid Username/Account Number (contains special characters)
     4. Invalid Username/Account Number (length exceeds maximum allowed)
  2. Password
     1. Valid Password
     2. Invalid Password (empty)
     3. Invalid Password (less than the required minimum length)
     4. Invalid Password (missing uppercase letter)
     5. Invalid Password (missing lowercase letter)
     6. Invalid Password (missing digit)
     7. Invalid Password (missing special character)

## **Test Cases:**

Banking Application (Bank Hub)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC ID | Scenarios | Pre requisites | Test data | Test steps | Expected result | Actual result | Status |
| BH-1 | Successful Login | User already registered | Kamran | User must navigate to the login page | Successful login  user can use the bank application |  |  |
| BH-2 | Invalid username (empty) | User registered | “” | Must navigate to login page | Display an error message indicating that the username/account number is required. |  |  |
| BH-3 | Invalid Username (contains special characters) | User registered | kamran@ | Must navigate to login page | Display an error message indicating that username containing special characters. |  |  |
| BH-4 | Invalid Username (length exceeds maximum) | User registered | Kamrankhan123alikamal001 | Must navigate to login page | Display an error message indicating that username exceeded the max length |  |  |
| BH-5 | Invalid Password (empty) | User registered |  | Must navigate to login page | Display an error message indicating that password is required. |  |  |
| BH-6 | Invalid Password (below minimum length) | User registered |  | Must navigate to login page | Display an error message indicating that password is below the min length |  |  |
| BH-7 | Invalid Password (missing uppercase letter) | User registered |  | Must navigate to login page | Display an error message indicating that password is missing the special character |  |  |
| BH-8 | Invalid Password (missing lowercase letter) | User registered |  | Must navigate to login page | Display an error message indicating that password is missing the lowercase letter |  |  |
| BH-9 | Invalid Password (missing digit) | User registered |  | Must navigate to login page | Display an error message indicating that password is missing the digit |  |  |
| BH-10 | Invalid Password (missing special character) | User registered |  | Must navigate to login page | Display an error message indicating that password is missing the special character |  |  |

# **Boundary Value Analysis**

Boundary Value Analysis (BVA) is a software testing technique used to identify test cases that focus on the boundaries of input domains.

## **Boundary Value Analysis for login functionality:**

Assumption:

1. The minimum length for the username and password is 6 characters.
2. The maximum length for the username and password is 20 characters.
3. The application only accepts alphanumeric characters for the username and password.

## **Boundary values for input parameter:**

* Username Length:
* Minimum length: 6 characters
* Maximum length: 20 characters
* Password Length:
* Minimum length: 6 characters
* Maximum length: 20 characters
* Username and Password Content:
* Alphanumeric characters (letters and numbers)

## **Test cases focusing on boundary values:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC ID | Scenario | Pre requisites | Test data | Test steps | Expected result | Actual result | status |
| BH-1 | Login with Minimum Valid Username Length | User exists | Kamran  Password  [valid] | 1)Navigate to login page  2)Enter username  3)Enter password  4)Click on the login button | Successful login |  |  |
| BH-2 | Login with Maximum Valid Username Length | User exists | Kamranaliquraishi  Password [valid] | same as above | Successful login |  |  |
| BH-3 | Login with Username Length < Minimum (Expecting Error) | User exists | Kamra  Password [valid] | same as above | Error message indicating a minimum length violation |  |  |
| BH-4 | Login with Username Length > Maximum (Expecting Error) | User exists | Kamranaliquraishifromkarachihai  Password [valid] | same as above | Error message indicating a maximum length violation |  |  |
| BH-5 | Login with Alphanumeric Username Within Valid Length | User exists | Kamran123  Password [valid] | same as above | Successful login |  |  |
| BH-6 | Login with Minimum Valid Password Length | User exists | Username [valid]  Pass123 | same as above | Successful login |  |  |
| BH-7 | Login with Maximum Valid Password Length | User exists | Username [valid]  Kamran12345678alii | same as above | Successful login |  |  |
| BH-8 | Login with Password Length < Minimum (Expecting Error) | User exists | Username [valid]  pass | same as above | Error message indicating a minimum length violation |  |  |
| BH-9 | Login with Password Length > Maximum (Expecting Error) | User exists | Username [valid]  Passwordlengthismorethanthelength | same as above | Error message indicating a maximum length violation |  |  |
| BH-10 | Login with Alphanumeric Password Within Valid Length | User exists | Username [valid]  Kamran123 | same as above | Successful login |  |  |
| BH-11 | Login with Valid Username and Valid Password | User exists | Username [valid]  Password [valid] | same as above | Successful login |  |  |
| BH-12 | Login with Valid Username and Invalid Password | User exists | Username [valid]  Password [invalid] | same as above | Error message indicating invalid credentials |  |  |
| BH-13 | Login with Invalid Username and Valid Password | User exists | Username [invalid] password [valid] | same as above | Error message indicating invalid credentials |  |  |
| BH-14 | Login with Invalid Username and Invalid Password | User exists | Username [invalid] password [invalid] | same as above | Error message indicating invalid credentials |  |  |
| BH-15 | Login with Empty Username and Empty Password Fields | User does not exist | Username “”  password “” | same as above | Error message indicating empty fields |  |  |

# **Decision Tables**

## **Decision Table for Transferring Funds:**

|  |  |  |
| --- | --- | --- |
| No | Conditions | Actions |
| 01 | User is logged in | Allow fund transfer |
| 02 | Sufficient balance | Deduct transfer amount |
| 03 | Transfer amount within limit | Update sender's balance |
| 04 | Valid recipient account | Update recipient's balance |
| 05 | Sufficient session security | Log transaction |
| 06 | Transfer currency supported | Notify successful transfer |
| 07 | Sufficient transfer limit | Notify insufficient funds or limit exceeded |
| 08 | Recipient account exists | Verify that the recipient account exists in the system. |

## **Conditions:**

1. User is logged in:

Check if the user is logged into the online banking application.

1. Sufficient balance:

Ensure that the sender has enough balance for the transfer.

1. Transfer amount within limit:

Check if the transfer amount is within the allowed limit.

1. Valid recipient account:

Verify that the recipient's account is valid.

1. Sufficient session security:

Ensure that the session has the necessary security measures.

1. Transfer currency supported:

Confirm that the currency of the transfer is supported.

1. Sufficient transfer limit:

Check if the user has the authority to transfer the specified amount.

1. Recipient account exists:

Verify that the recipient account exists in the system.

## **Actions:**

1. Allow fund transfer:

Permit the user to initiate the fund transfer.

1. Deduct transfer amount:

Reduce the sender's balance by the transfer amount.

1. Update sender's balance:

Reflect the deduction in the sender's account.

1. Update recipient's balance:

Increase the recipient's balance by the transfer amount.

1. Log transaction:

Record the details of the transaction for security and auditing purposes.

1. Notify successful transfer:

Inform the user that the transfer was successful.

1. Notify insufficient funds or limit exceeded:

Alert the user if there are insufficient funds or the transfer limit is exceeded

## **Test cases based on decision table**

in all scenarios sender has $500 in balance

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC ID | Scenario | Pre requisites | Test data | Test steps | Expected result | Actual result | Status |
| BH-01 | Successful fund transfer | User must log in | recipient account valid | -Enter transfer amount within limit  -Select recipient account  -Confirm transfer | Fund transfer successful; Sender balance updated; Recipient balance updated |  |  |
| BH-02 | User not logged in | User not log in | Any | Attempt to initiate a fund transfer | User prompted to log in |  |  |
| BH-03 | Insufficient balance | User must log in | recipient account valid | Enter transfer amount exceeding the available balance | User notified of insufficient funds |  |  |
| BH-04 | Transfer amount exceeds limit | Same as above | recipient account valid | Enter transfer amount exceeding the allowed limit | User notified that the transfer amount exceeds the limit |  |  |
| BH-05 | Invalid recipient account | Same as above | invalid recipient account | Enter valid transfer amount | User notified that the recipient account is invalid |  |  |
| BH-06 | Insecure session | Same as above | recipient account valid | Tamper with the session security during the fund transfer process | Fund transfer aborted; User prompted to log in again |  |  |
| BH-07 | Unsupported transfer currency | Same as above | recipient account valid | Enter transfer amount in an unsupported currency | User notified that the currency is not supported |  |  |
| BH-08 | Exceeds transfer limit | Same as above | recipient account valid | Enter transfer amount exceeding the user's transfer limit | User notified that the transfer limit has been exceeded |  |  |
| BH-09 | Recipient account does not exist | Same as above | recipient account does not exist | Enter valid transfer amount | User notified that the recipient account does not exist |  |  |

# **References**

altexsoft. (2023, 11 30). *altexsoft*. Retrieved from altexsoft: https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/

Andrew. (2023, 03 23). *relevant*. Retrieved from relevant: https://relevant.software/blog/software-requirements-specification-srs-document/

geeksforgeeks. (n.d.). *geeksforgeeks*. Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/specification-based-testing/

geeksforgeeks. (n.d.). *geeksforgeeks*. Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/software-requirement-specification-srs-format/

javatpoint. (n.d.). *javatpoint*. Retrieved from javatpoint: https://www.javatpoint.com/software-requirement-specifications

lambdatest. (2023, 04 11). *lambdatest*. Retrieved from lambdatest: https://www.lambdatest.com/learning-hub/test-specification

perforce. (2023, 1 17). *perforce*. Retrieved from perforce: https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document

Rosencrance, L. (n.d.). *techtarget*. Retrieved from techtarget: https://www.techtarget.com/searchsoftwarequality/definition/software-requirements-specification

Sharma, L. (2021, 10 18). *toolsqa*. Retrieved from toolsqa: https://www.toolsqa.com/software-testing/specification-based-testing/

springer. (2020, 11 18). *springer*. Retrieved from springer: https://link.springer.com/article/10.1007/s11219-020-09523-0

Thomas, A. T. (2023, 8 2). *testsigma*. Retrieved from testsigma: https://testsigma.com/blog/specification-based-testing/