K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Batch: A2 **Roll No.: 16010121045**

Experiment / assignment / tutorial No.___7_

Title: Designing test plan document for Mini Project

Aim: To learn and understand the way of developing the software by classical methods of software engineering. Planning and monitoring, testing, validating of the project using tools and prepares a document for the same by using the concept of software engineering

---

CO:

Books/ Journals/ Websites referred:

1. Roger Pressman, Software Engineering: A practitioners Approach, McGraq Hill, 2010 ,6th edition
2. Ian Somerville , Software Engineering , Addison Wesley,2011,9th edition
3 http://en.wikipedia.org/wiki/Software_requirements_specification

---

Test Plan Template:

CodeCat- The online Complier

Prepared by:

Meet Gala

PargatSingh Dhanjal

Vishrut Deshmukh

10/5/2023

TABLE OF CONTENTS

1.0 INTRODUCTION

2.0 OBJECTIVES AND TASKS
- 2.1 Objectives
- 2.2 Tasks

3.0 SCOPE

4.0 Testing Strategy
4.1 Alpha Testing (Unit Testing)
4.2 System and Integration Testing
4.3 Performance and Stress Testing
4.4 User Acceptance Testing
4.5 Batch Testing
4.6 Automated Regression Testing
4.7 Beta Testing

5.0 Hardware Requirements

6.0 Environment Requirements
6.1 Main Frame
6.2 Workstation

7.0 Test Schedule

8.0 Control Procedures

9.0 Features to Be Tested

10.0 Features Not to Be Tested

11.0 Resources/Roles & Responsibilities

12.0 Schedules

13.0 Significantly Impacted Departments (SIDs)

14.0 Dependencies

15.0 Risks/Assumptions

16.0 Tools

17.0 Approvals

1.0 INTRODUCTION

Brief Summary of the Product Being Tested:

The product under testing is "CodeCat – Online Compiler," an online coding platform designed to offer an all-inclusive environment for programmers to write, compile, and execute code in multiple programming languages. It addresses the challenges faced by students and developers when using online compilers. The core functions of CodeCat include:

High-Level Overview of Functions:

Multi-Language Support: CodeCat supports over 30 programming languages, providing users with the flexibility to choose the language that best suits their needs.

Syntax Highlighting: The platform offers syntax highlighting for most popular programming languages, making the coding experience akin to working in a dedicated integrated development environment (IDE).

Error and Success Alerts: Users receive immediate feedback on their code through error and success alerts, ensuring that they can quickly identify and rectify issues.

Real-Time Compilation and Execution: CodeCat enables users to compile and execute their code directly within the platform, eliminating the need for additional software or tools.

Multi-Line Input/Output Windows: The platform provides multi-line input/output windows for users to interact with their code and view results effectively.

Light and Dark Mode: CodeCat offers both light and dark mode options, allowing users to customize their working environment to their preference.

Progressive Web Application (PWA): Users can access CodeCat on a wide range of devices, including desktop and mobile, thanks to its PWA capabilities.

2.0 OBJECTIVES AND TASKS

2.1 Objectives:

Testing Framework Definition: The primary objective is to establish a comprehensive testing framework for CodeCat, outlining methodologies and standards.

Core Functionality Validation: Ensure that CodeCat's core features, like syntax highlighting and code compilation, meet their intended purpose.

Quality Assurance: Set quality criteria and performance benchmarks to maintain high product quality.

Effective Communication: Act as a central document for team communication, clarifying roles, responsibilities, and timelines.

Service Level Agreement (SLA): Define the agreed-upon standards and benchmarks to maintain during testing, aligning expectations.

2.2 Tasks:

Functional Testing: Assess the core functions of CodeCat, including multi-language support, syntax highlighting, error handling, and compilation.

UI/UX Testing: Evaluate the user interface for usability, responsiveness, and visual appeal.

Security Testing: Identify vulnerabilities and ensure the security of user data.

Performance Testing: Evaluate response times and scalability to meet user demands.

Compatibility Testing: Test CodeCat's compatibility with various devices and web browsers to ensure a seamless user experience.

3.0 SCOPE

General : The scope of the testing effort for CodeCat encompasses:

- Testing all functions and features of the CodeCat platform.
- Evaluating the user interface and user experience (UI/UX).
- Testing existing interfaces and ensuring seamless integration of all functions.
- Assessing the compatibility of CodeCat with various devices and web browsers.

Tactics: To accomplish the items listed in the "Scope" section, the following tactics will be employed:

Testing All Functions: A comprehensive testing strategy will be developed to cover all the functions of CodeCat. This includes syntax highlighting, code compilation, error handling, and multi-language support.

UI/UX Evaluation: A user interface and user experience assessment will be conducted, involving usability testing and feedback from real users to ensure the platform is user-friendly.

Testing Existing Interfaces: The testing team will collaborate with key stakeholders to identify and test existing interfaces, ensuring that all interactions are smooth and error-free.

Compatibility Assessment: CodeCat will be tested across a variety of devices, operating systems, and web browsers to verify its compatibility and usability. Any compatibility issues will be documented and addressed.

Stakeholder Collaboration: Key individuals representing different areas, such as development, UI/UX design, and quality assurance, will be informed and included in the testing process. This will require scheduling and allocating time for their participation to ensure a well-rounded evaluation.

User Feedback: Real users will be invited to provide feedback on the platform's performance and usability, and this feedback will be incorporated into the testing process.

Documentation: Clear and concise documentation of the testing scope and procedures will be maintained and communicated to all relevant stakeholders to ensure transparency and alignment.

4.0 TESTING STRATEGY

The strategy for testing is to break down the application into small units and test each unit to ensure proper functioning of each unit. The team also plans to integrate the small units and perform testing on the integrated system to ensure that there is smooth functioning of the application as a whole. Testing strategies planned to be used:

- Unit Testing
- System and Integration Testing
- Performance and Stress testing
- User Acceptance Testing
- Batch testing
- Beta Testing

4.1 Unit Testing

Definition:

Unit testing validates individual code units within CodeCat, ensuring they perform their intended tasks correctly. It aims for comprehensive coverage, error minimization, and thorough requirement tracing.

Participants:

- Meet Gala
- Pargat Singh Dhanjal
- Vishrut Deshmukh

Methodology:

- Develop test scripts with input data and expected outcomes.
- Execute unit tests systematically for each unit.
- Identify and document defects, assess their severity.
- Resolve defects, retest units for validation.
- Analyze code coverage for thorough testing.
- Maintain requirement traceability.

Completion criteria: High code coverage, defect resolution, and requirement coverage.

4.2 System and Integration Testing

Definition:
System and Integration Testing for CodeCat involves evaluating the combined functionality of multiple units, modules, or components, ensuring that they interact correctly and that the entire system functions as a unified whole. It tests end-to-end scenarios, data flow, and identifies any issues related to integrated components or external interfaces.

Participants:

- Meet Gala
- Pargat Singh Dhanjal
- Vishrut Deshmukh

Methodology:
System and Integration Testing will follow this methodology:

Test Scenario Design: Test scenarios will be developed to evaluate various end-to-end user interactions and system behavior. These scenarios will encompass the integration of different modules and components.

Test Script Development: The Quality Assurance Team will create test scripts based on the defined test scenarios. These scripts will specify the sequence of actions, input data, and expected

4.3 Performance and Stress Testing

Definition:
Performance and Stress Testing for CodeCat is the evaluation of the application's responsiveness, stability, and scalability under different loads and stressful conditions. It assesses how well the system performs under normal usage and extreme conditions, including high user loads and resource limitations.

Participants:

- Meet Gala
- Pargat Singh Dhanjal
- Vishrut Deshmukh

Methodology:
Test Scenario Design: The Quality Assurance Team will define test scenarios to assess the application's performance under normal conditions, as well as scenarios that push the system to its limits.

- Test Script Development: Test scripts will be created to simulate user interactions and load scenarios. These scripts will specify the sequence of actions, input data, and expected outcomes.
- Performance Testing: Tests will be executed under normal usage conditions to evaluate the application's responsiveness, throughput, and resource utilization.
- Stress Testing: The application will be subjected to extreme conditions, such as high user loads, limited resources, or simulated attacks, to assess its stability and scalability.
- Bottleneck Identification: Performance bottlenecks will be identified and analyzed. The development team will work on optimizing the application based on the findings.
- Resource Monitoring: Various resources like CPU, memory, and network utilization will be monitored to detect potential issues or overloads.
- Completion Criteria: Performance and Stress Testing will be considered complete when the application demonstrates stable performance under expected loads, and potential bottlenecks have been addressed.

4.4 User Acceptance Testing

Definition:
The purpose of the acceptance test is to confirm that the system is ready for operational use. During the acceptance test, end-users (customers) of the system compare the system to its initial requirements. UAT is done in the final phase of testing after functional, integration and system testing are done.

Participants:

Meet Gala

Pargat Singh Dhanjal

Vishrut Deshmukh

Methodology:
User Acceptance Testing will follow this methodology:

Test Scenario Identification: UAT test scenarios will be designed to cover various user interactions, including code compilation, execution, and platform usability. These scenarios will align with the system's initial requirements.

Test Script Development: The end-users (Meet, Pargat, and Vishrut) will create UAT test scripts based on the defined test scenarios. These scripts will specify the steps they will perform, the expected outcomes, and any issues they encounter.

UAT Execution: Each participant will execute their respective test scripts, verifying the system's functionality and user-friendliness.

Issue Reporting: If any discrepancies are identified between expected and actual outcomes, the participants will report these issues. The severity and impact of these issues will be documented.

Resolution and Retesting: The development team will address reported issues and defects. Subsequently, the participants will retest the system to confirm issue resolution.

User Feedback: Participants will provide feedback on the overall user experience and suggest improvements or enhancements.

Completion Criteria: User Acceptance Testing will be considered complete when the participants confirm that the system aligns with their initial requirements and is ready for operational use.

4.5 Batch Testing

Definition:
A batch test occurs when you run multiple scripts. It is typically done with automation. You program a batch test by placing the scripts in the order you wish to have them run and employing a tool that will execute the scripts in that specified order.

Participants:

Meet Gala

Pargat Singh Dhanjal

Vishrut Deshmukh

Methodology:

- Select Test Scripts: Choose diverse test scripts representing different scenarios.
- Define Execution Order: Specify the order in which test scripts will run.
- Automate with a Tool: Configure an automation tool to execute scripts sequentially.
- Execute Batch Test: Run selected scripts in the defined order with automation.
- Document Issues: Record discrepancies, errors, or defects encountered.
- Resolve and Re-Test: Address issues and re-run batch tests if necessary.
- Evaluate Performance: Assess system performance under continuous usage.
- Completion Criteria: Testing is complete when all scripts run successfully, and issues are resolved.

4.6 Automated Regression Testing

Definition:
Regression testing is the selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still works as specified in the requirements. Automated Regression testing is a black-box testing technique that consists of re-executing those tests that are impacted by the code changes. These tests should be executed as often as possible throughout the software development life cycle.

Participants:

Meet Gala

Pargat Singh Dhanjal

Vishrut Deshmukh

Methodology:

Test Suite Identification: Select and maintain a comprehensive test suite that covers all critical aspects of the CodeCat system. This suite should include test cases for both new features and existing functionalities.

Test Automation: Utilize automated testing tools and frameworks to create and execute regression test scripts. These scripts will test the application's functionality in a repeatable and consistent manner.

Version Control Integration: Ensure that the automated regression testing suite is integrated with the version control system. This enables easy tracking of changes in the codebase and triggers automatic testing when modifications are made.

Scheduled Testing: Set up scheduled or trigger-based automated regression tests. When code changes are committed or before new releases, the automated tests are executed.

Result Analysis: Review the test results to identify any discrepancies, errors, or failures compared to the baseline results. These discrepancies are flagged as potential regression defects.

Defect Reporting: Report identified regression defects to the development team for resolution. The severity and impact of each defect are documented.

Issue Resolution: Developers address and resolve reported defects, ensuring that the codebase aligns with the intended functionality.

4.7 Beta Testing

Participants:

Meet Gala

Pargat Singh Dhanjal

Vishrut Deshmukh

Methodology:

We plan on taking feedback from a few of our fellow classmates with the help of Google forms. The form shall consist of various questions that will help us understand the usability and reliability of the application from the user’s point of view. Any constructive suggestions will be worked upon in the future scope of the application. We also will ask our lab teacher for inputs on how the application can be improved and better presented.

5.0 HARDWARE REQUIREMENTS

Hardware	Configuration	No. of units
Computers	Intel Core 5, 8GB RAM, 50GB Hard disk.	1

6.0 ENVIRONMENT REQUIREMENTS

A suitable browser with active internet connection.

6.1 Main Frame

Specify both the necessary and desired properties of the test environment. The specification should contain the physical characteristics of the facilities, including the hardware, the communications and system software, the mode of usage (for example, stand-alone), and any other software or supplies needed to support the test. Also specify the level of security which must be provided for the test facility, system software, and proprietary components such as software, data, and hardware.

Identify special test tools needed. Identify any other testing needs (for example, publications or office space). Identify the source of all needs which are not currently available to your group.

6.2 Workstation

7.0 TEST SCHEDULE

Duration	Milestone	Output
1st week	Unit testing	Documentation and feedback.
2nd week	System integration testing	Developer’s comments, automated tool reports.
3rd week	Performance testing and User Acceptance testing	User’s feedback, system performance report
4th week	Beta testing	User’s feedback

8.0 CONTROL PROCEDURES

Problem Reporting

Document the procedures to follow when an incident is encountered during the testing process. If a standard form is going to be used, attach a blank copy as an “Appendix” to the Test Plan. In the event you are using an automated incident logging system, write those procedures in this section.

- If any problem or error has occurred during the testing, the tester can directly report the error to the developer, also the developer makes note of errors to be corrected

Change Requests

Document the process of modifications to the software. Identify who will sign off on the changes and what would be the criteria for including the changes to the current product. If the changes will affect existing programs, these modules need to be identified.

If any error is found by the tester, and if it can be resolved and 70% of developer teams agree to make the changes, then the changes are to be adapted in necessary modules

9.0 FEATURES TO BE TESTED

Identify all software features and combinations of software features that will be tested.

Login	If user can log into the system.
Registration and Github Link	If new users can register and sync is github account
Code writing , syntax	If the user can write code effectively with proper syntax sugestions
Theme check	Check if both light and dark mode are working with proper design color theme
Output generation	Check if the api is connected and rendered and backend to get the output showed successfully.
Error generation	Check if users are notified with errors properly

10.0 FEATURES NOT TO BE TESTED

Identify all features and significant combinations of features which will not be tested and the reasons.

Landing Page

11.0 RESOURCES/ROLES & RESPONSIBILITIES

Meet Gala: Prepataion of Slides, resolving test cases
Pargat Singh Dhanjal : Preparing, executing and resolving test cases.
Vishrut Deshmukh : Managing, designing test cases.

12.0 SCHEDULES

Major Deliverables
Identify the deliverable documents. You can list the following documents:
– Test Plan
– Test Cases
– Test Incident Reports
– Test Summary Reports

13.0 SIGNIFICANTLY IMPACTED DEPARTMENTS (SIDs)

Department/Business Area Bus. Manager Tester(s)

SID	Department In-charge
Output generation and api integartion	Pargat Singh Dhanjal
Website testing	Meet Gala
Quality Assurance	Vishrut Deshmukh

14.0 DEPENDENCIES

Identify significant constraints on testing, such as test-item availability, testing-resource availability, and deadlines.

Since there are many modules and computer languages integrated in the website, time will be a major constraint to carry out testing for each and every module.

15.0 RISKS/ASSUMPTIONS

It might take a extra time to return the output as many calls are made to api or the network is not stable.

16.0 TOOLS

No specific external automation or bug tracking tools are being used for this website.

17.0 APPROVALS

Specify the names and titles of all persons who must approve this plan. Provide space for the signatures and dates.

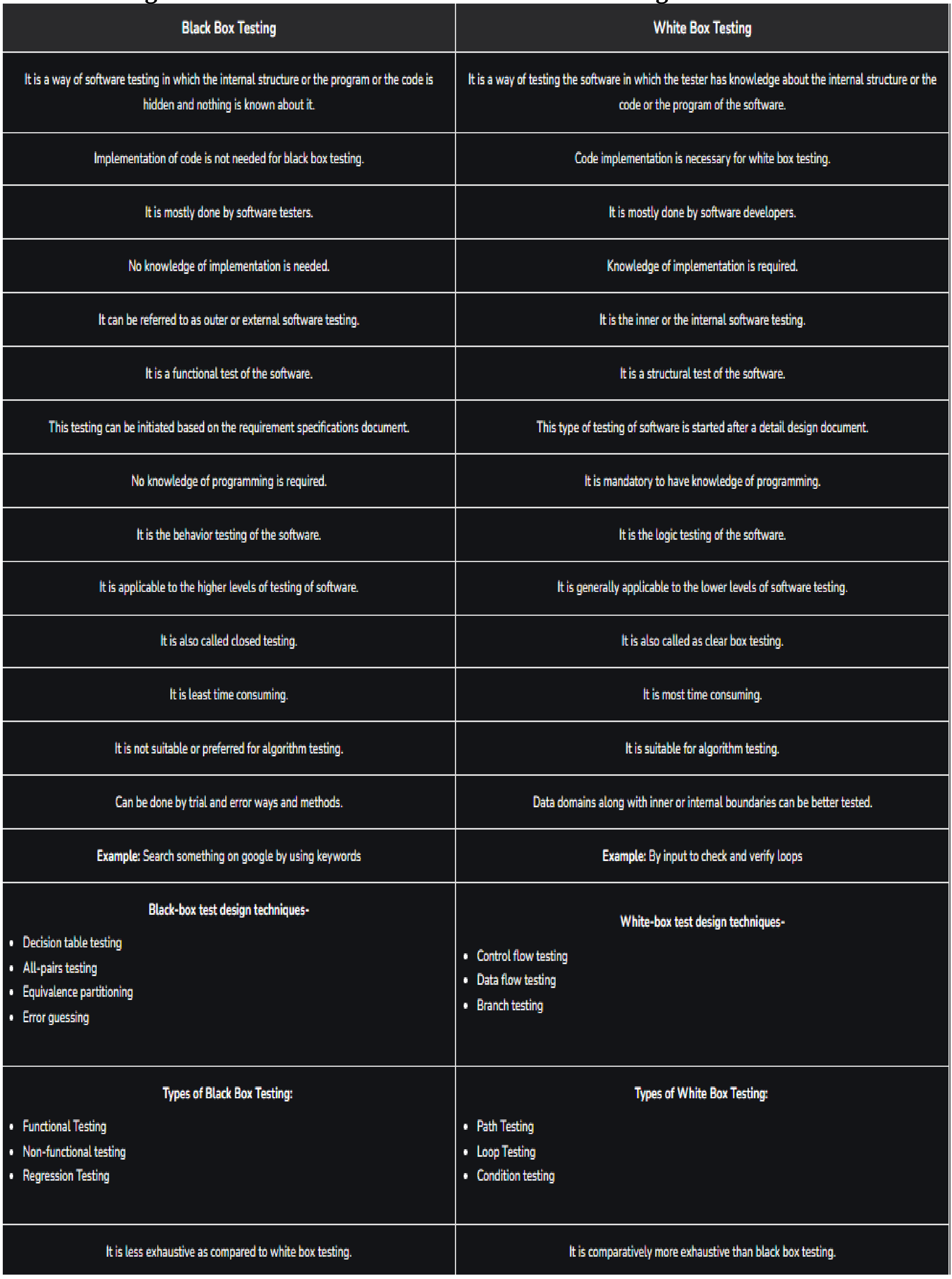
Name (In Capital Letters)

Signature

Date

Post Lab Descriptive Questions:

1. **Distinguish between Black Box and White Box Testing**



Black Box Testing	White Box Testing
It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.
Implementation of code is not needed for black box testing.	Code implementation is necessary for white box testing.
It is mostly done by software testers.	It is mostly done by software developers.
No knowledge of implementation is needed.	Knowledge of implementation is required.
It can be referred to as outer or external software testing.	It is the inner or the internal software testing.
It is a functional test of the software.	It is a structural test of the software.
This testing can be initiated based on the requirement specifications document.	This type of testing of software is started after a detail design document.
No knowledge of programming is required.	It is mandatory to have knowledge of programming.
It is the behavior testing of the software.	It is the logic testing of the software.
It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
It is also called closed testing.	It is also called as clear box testing.
It is least time consuming.	It is most time consuming.
It is not suitable or preferred for algorithm testing.	It is suitable for algorithm testing.
Can be done by trial and error ways and methods.	Data domains along with inner or internal boundaries can be better tested.
Example: Search something on google by using keywords	**Example:** By input to check and verify loops
Black-box test design techniques- • Decision table testing • All-pairs testing • Equivalence partitioning • Error guessing	**White-box test design techniques-** • Control flow testing • Data flow testing • Branch testing
Types of Black Box Testing: • Functional Testing • Non-functional testing • Regression Testing	**Types of White Box Testing:** • Path Testing • Loop Testing • Condition testing
It is less exhaustive as compared to white box testing.	It is comparatively more exhaustive than black box testing.

2. Consider following scenario: An institute is interested in developing a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

   The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (e.g. passwords) is stored in plain text.

Assume following test suite is used

A test suite to verify the "User Login" feature

#	TS1
Title	**Verify "User Login" functionality**
Description	**To test the different scenarios that might arise while an user is trying to login**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
TC1	Verify that user already registered with the LIS is able to login with correct user ID and password		Employee ID *149405* is a registered user of LIS; user's password is *this_is_password*	User is logged in	1. Type in employee ID as *149405* 2. Type in password *this_is_password* 3. Click on the 'Login' button	"Home" page for the user is displayed
TC2	Verify that an unregistered user of LIS is unable to login		Employee ID *149405xx* is not a registered user of LIS	User is not logged in	1. Type in employee ID as *149405xx* 2. Type in password *whatever* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message
TC3	Verify that user already registered with the LIS is unable to login with incorrect password		Employee ID *149405* is a registered user of LIS; user's password is *this_is_password*	User is not logged in	1. Type in employee ID as *149405* 2. Type in password *whatever* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message

A test suite to verify the "User Login" feature	
#	**TS1**
Title	**Verify "User Login" functionality**
Description	**To test the different scenarios that might arise while an user is trying to login**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
TC4	Verify that user already registered with the LIS is unable to login with incorrect password given twice consecutively	TC3	This test case is executed after execution of TC3 before executing any other test case	User is not logged in	1. Type in employee ID as *149405* 2. Type in password *whatever2* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message
TC5	Verify that user already registered with the LIS is unable to login with incorrect password given thrice consecutively	TC4	This test case is executed after execution of TC4 before executing any other test case	User is not logged in	1. Type in employee ID as *149405* 2. Type in password *whatever3* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message; the security question and input box for the answer are displayed
TC6	Verify that a registered user can login after three consecutive failures by correctly answering the security question	TC5	This test case is executed after execution of TC6 before executing any other test case. Answer to the security question is *my_answer*.	Email sent containing new password. The email is expected to be received within 2 minute.	1. Type in the answer as *my_answer* 2. Click on the 'Email Password' button	Login dialog is displayed; an email containing the new password is received

A test suite to verify the "User Login" feature

#	**TS1**
Title	**Verify "User Login" functionality**
Description	**To test the different scenarios that might arise while an user is trying to login**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
TC7	Verify that a registered user's account is blocked after three consecutive failures and answering the security question incorrectly		Execute the test cases TC3, TC4, and TC5 once again (in order) before executing this test case	User account has been blocked	1. Type in the answer as *not_my_answer* 2. Click on the 'Email Password' button	The message *"Your account has been blocked! Please contact the administrator."* appears

create a Requirements Traceability Matrix (RTM) showing a mapping from individual requirement to test case(s).

Table 1: A simplified mapping from requirements to test cases

Requirement #	Test Case #
R1	TC1
R2	TC2
R3	TC3
R4	TC4

Consider requirements are summarized in the table below

#	Requirement
R1	New user registration
R2	User Login
R3	Search book
R4	Issue book
R5	Return book
R6	Reissue book