CEBU INSTITUTE OF TECHNOLOGY

UNIVERSITY

COLLEGE OF COMPUTER STUDIES

Software Test Document

for

CyberKids

Change History

Version	Date	Amendment	Author
1.0	9/5/2025	Initial	Cultura

Table of Contents

1. Introduction **6**

1.1. System Overview

1.2. Test Approach 6

1. Introduction

1.1 System Overview

The **CyberKids** platform is an interactive, educational game-based system developed to promote cybersecurity awareness among elementary school students, specifically those in **Grades 5 to 6**. It integrates a Roblox-based game client, a web-based teacher dashboard, and a secure backend server to deliver a cohesive and engaging cybersecurity learning experience.

The system is composed of three major components, each with distinct responsibilities yet integrated through RESTful APIs and real-time communication mechanisms:

1. Roblox Game Client

The front-facing component for students is developed in Roblox Studio using Lua scripting, consisting of three core levels representing different cybersecurity themes:

- Online Privacy (Drag-and-drop classification game)
- Password Security (Password construction game)
- Phishing Awareness (Escape-room style phishing detection)

Each level incorporates a scoring system, timer, and leaderboard to gamify learning and maintain engagement. The game communicates with the backend server through secure API calls to submit scores, retrieve leaderboard data, and log progress.

2. Backend API Server

The backend is developed using **Spring Boot (Java)** and is responsible for:

- Handling **student registration**, including secure storage of real names
- Managing **leaderboard calculations**
- Storing and retrieving **game scores**
- Broadcasting **real-time updates** to the teacher dashboard
- Enforcing **data encryption and access control**

3. Teacher Dashboard

The teacher dashboard is a ReactJS-based web application that allows educators to:

- Log in using Microsoft Teams credentials via OAuth
- View real-time student progress and activity
- Manage access to game levels (lock/unlock)
- Create and organize class sections
- Export performance reports for assessment purposes
- Receive real-time notifications for registration, completion, and online status

The dashboard is synchronized with the backend server using a combination of polling and WebSocket-based real-time updates, ensuring that educators have up-to-date visibility into their students' activities.

Overall, the CyberKids platform follows a modular, loosely coupled architecture that allows independent development and testing of each component while ensuring seamless integration and communication across the system.

1.2 Test Approach

The testing strategy for CyberKids adheres strictly to an **Agile iterative methodology**, emphasizing continuous testing, feedback integration, and feature-driven validation. Given the evolving nature of educational technology projects—especially those targeted at young learners—Agile offers the necessary flexibility to incorporate educator feedback, usability adjustments, and performance optimizations throughout the development lifecycle.

Agile Testing Methodology

Testing is conducted in short iterations (sprints) aligned with the Agile development cycle. At the end of each sprint, newly implemented features are subjected to multiple layers of testing before being accepted into the working product. This approach supports incremental delivery of high-quality, thoroughly validated features.

Key testing activities under this approach include:

1. Sprint-Based Functional Testing

After each development sprint, newly added features—such as a new game level or dashboard functionality—are tested against defined acceptance criteria. This ensures that each feature performs as intended before it is integrated into the live environment.

- Example: After implementing the Password Security Level, tests are executed to verify scoring accuracy, password strength validation, and leaderboard updates.

2. Continuous Integration and Testing

New builds are regularly pushed to a staging environment, where a set of basic smoke tests are executed to confirm the stability of the system. This includes:

- Backend API availability
- Frontend rendering
- Game startup and level loading
- Real-time communication between components

3. Developer and Peer Reviews

Before merging new code, developers conduct unit tests locally and submit code for peer review. This practice helps identify logical errors, improve test coverage, and maintain coding standards across the team.

4. Cross-Component Integration Testing

Integration tests are executed to ensure that the system components—Roblox client, backend server, and React dashboard—communicate correctly and maintain data consistency.

Key integration test cases include:

- Submitting a score from the game and verifying its appearance on the teacher dashboard
- Registering a new student and confirming encrypted data storage
- Locking/unlocking levels from the dashboard and observing real-time effects in the game

5. Usability Testing within Iterations

Given the young target audience, each major release includes basic usability testing sessions with elementary students and feedback sessions with partner teachers. These are conducted in controlled settings, and insights gathered are used to refine UI/UX elements in the following sprint.

6. Regression Testing

At the end of every sprint cycle, a set of predefined regression tests are run to verify that recent code changes have not introduced bugs into previously working features. These cover critical functions such as:

- Score tracking
- Leaderboard updates
- Student login and registration
- Dashboard real-time updates

7. Acceptance Testing (User Validation)

Before major deployment or presentation milestones, a formal User Acceptance Testing (UAT) phase is conducted involving the project adviser and cooperating teachers. This phase is guided by high-level user stories such as:

- “As a teacher, I want to monitor student progress in real time.”
  “As a student, I want to see my ranking on the leaderboard after finishing a game.”

Success is determined based on whether the system behavior aligns with expected outcomes from the end-user perspective.

8. Documentation and Feedback Loop

Every testing phase includes the following documentation:

- Test case reports
- Issue and bug logs
- Sprint retrospectives
- Change logs

Identified issues are documented and prioritized in the product backlog for resolution in the next sprint.

1.3 Definitions and Acronyms

This section provides definitions of key terms, abbreviations, and acronyms used throughout the Software Test Document to ensure consistency and clarity.

Term / Acronym	Definition
Agile Methodology	A software development approach based on iterative development, where requirements and solutions evolve through collaboration between cross-functional teams.
API (Application Programming Interface)	A set of protocols and tools that allows different software components to communicate with each other.
Backend	The server-side part of the system responsible for data processing, storage, and business logic. In CyberKids, this includes score handling, user registration, and real-time event broadcasting.
Client	The front-end interface used by end users. In CyberKids, the game client used by students and the dashboard used by teachers both function as clients.
Cybersecurity	The practice of protecting systems, networks, and programs from digital attacks, especially those intended to access, change, or destroy sensitive information.
Dashboard	A web-based user interface developed in ReactJS for teachers to monitor student activity, view scores, manage levels, and access real-time updates.
Encryption	The process of converting information into a coded format to prevent unauthorized access. CyberKids encrypts student real names to protect their identity.
OAuth (Open Authorization)	A secure authorization protocol that allows users to log into third-party applications using their credentials from platforms like Microsoft Teams.

2. Test Plan

2.1 Features to be Tested

Feature	Description	Priority
Level 1 Game: Information Classification Sorting Challenge	Verifies drag-and-drop functionality, scoring logic, and leaderboard update after level completion.	High
Level 2 Game: Password Security Challenge	Ensures password creation mechanics work, timers run accurately, and scoring reflects password strength.	High
Level 3 Game: Phishing Identification Challenge	Validates correct point awarding/deduction based on phishing flag actions and completion logic.	High
Timer System	Confirms timers start/stop correctly and integrate with scoring and leaderboard systems.	Medium
Scoring System	Ensures scores are calculated accurately based on user actions in each level.	Medium
Leaderboard System	Verifies leaderboard ranks students correctly based on score and completion time.	Medium
Student Registration System	Tests registration flow, Roblox ID linking, and real-name encryption in database.	High
Teacher Dashboard: Lock/Unlock Levels	Ensures teachers can control access to specific game levels for each student or class.	Low

Teacher Dashboard: Reassign Students to Levels	Validates ability to reassign students to different levels manually.	Low
Teacher Dashboard: Class Creation and Enrollment	Verifies ability to create class sections and enroll students under grade levels and sections.	Low
Microsoft Teams Authentication	Tests login functionality using OAuth for teacher accounts and secure access control.	Low
Real-Time Notification System	Ensures notifications are triggered for key student events (registration, level completion, etc.) and received by teachers.	High
Online/Offline Status Tracking	Confirms teacher dashboard correctly shows students' current online/offline status.	Low

2.2 Features not to be Tested

The following features and aspects are excluded from the current scope of testing due to project constraints, external dependencies, or lower testing priority. Their exclusion does not impact the verification of core functionality.

Feature/Aspect	Reason for Exclusion
Cross-browser and cross-device compatibility	Testing is limited to Microsoft Edge on desktop. Other browsers (Chrome, Firefox, Safari) and mobile devices are out of scope for this testing cycle.
Backend performance and load testing	Stress, scalability, and endurance tests of APIs are excluded due to limited resources; only functional correctness is validated.
Microsoft Teams token refresh and long-term authentication scenarios	The integration relies on Microsoft's maintained OAuth service, which is assumed to handle extended sessions and token expiration correctly.
Rare concurrency issues (e.g., two teachers modifying the same student assignment simultaneously)	Out of scope for current testing, as such cases are unlikely under normal use.
Cosmetic UI aspects (colors, themes, animations, layout responsiveness)	Focus is on functional correctness; minor UI design issues do not affect core functionality.

2.3 Testing Tools and Environment

The testing activities for the CyberKids project were conducted using a focused set of tools aligned with the system's architecture and technology stack. Given the project's cross-platform nature (Roblox for the game, Spring Boot backend, and React for the dashboard), testing was performed across multiple environments to ensure functional integrity, security, and usability.

Testing Tools

- **Postman** was used to manually test RESTful API endpoints between the Roblox client, backend server, and teacher dashboard, ensuring correct data handling and response formats.
- **Roblox Studio** served as the development and testing environment for all in-game functionality, including scripting, UI behavior, API integration, and level progression.
- **Microsoft Edge** was used to test the ReactJS-based teacher dashboard for UI responsiveness, real-time updates, and Microsoft Teams OAuth authentication.

Testing was conducted in both local and live environments to simulate real-world use, with a focus on compatibility, functionality, and system integration across components.

Testing Environment

- **Backend Server**: Deployed on a local or cloud-based Spring Boot instance using Java. Tested with mock and real data sets.
- **Database**: Used during backend testing to verify secure storage of student data and leaderboard metrics.
- **Network Conditions**: Testing included both offline/local scenarios and live internet-connected testing to ensure reliability in real-world classroom settings.
- **Devices**: Tests were performed on standard development laptops, with Roblox game testing also executed on compatible mobile devices to verify cross-device compatibility.

3. Test Cases

3.1 Case - 1: Level 1 Game: Information Classification Sorting Challenge

3.1.1 Purpose

To verify the full functionality of Level 1 (Online Privacy) game, including drag-and-drop interaction, scoring, timer, leaderboard update, and dynamic content loading from the teacher dashboard. This also ensures that the teacher's custom input (information items) is successfully retrieved and rendered in the Roblox game.

3.1.2 Inputs

Input Type	Description
Timer Trigger	Timer starts automatically when the level begins
Teacher Configures & Game Fetches Items	Teacher inputs a custom set of information items via the dashboard, and the Roblox client dynamically fetches and displays them in-game
User Action – Correct Sort	User drags a correct item to the correct bin (e.g., "Nickname" → Safe to Share)
User Action – Incorrect Sort	User drags an incorrect item to the wrong bin (e.g., "Home Address" → Safe)
Timer Completion Trigger	All items are sorted; level is completed
Score Submission Trigger	Final score and time are sent to backend after level completion
Leaderboard Update Trigger	System updates leaderboard with new score and completion time

3.1.3 Expected Outputs & Pass/Fail Criteria

Related Input	Expected Output	Pass/Fail Criteria
Timer Trigger	Timer starts counting down/up as soon as level begins	**Pass** if timer starts immediately; **Fail** if delayed or doesn't start
Teacher Configures & Game Fetches Items	Teacher-defined classification items are saved and successfully fetched & displayed in Roblox	**Pass** if items appear correctly; **Fail** if missing or incorrect
User Action – Correct Sort	+Points added to score	**Pass** if points increase correctly; **Fail** if no points added or incorrect
User Action – Incorrect Sort	No points added to score	**Pass** if no points awarded for wrong sort; **Fail** if points added incorrectly
Timer Completion Trigger	Timer stops and total time is recorded	**Pass** if timer stops on completion; **Fail** if timer keeps running or stops incorrectly
Score Submission Trigger	Score and time sent to backend via REST API	**Pass** if score/time sent correctly; **Fail** if not sent or incorrect data
Leaderboard Update Trigger	Player's score and time appear on leaderboard	**Pass** if leaderboard updates correctly; **Fail** if not updated or incorrect

3.1.4 Test Procedure

Step No.	Test Action	Expected Result
1	Teacher logs in to dashboard and configures a new set of classification items	New items are saved in database
2	Launch the game and select **Online Privacy** level	Level 1 loads; game environment initializes
3	Timer begins as game starts	Timer starts counting
4	Start Level 1 in Roblox and verify that configured items are displayed	Timer starts counting Game fetches and shows teacher-defined items
5	Student correctly sorts item (e.g., "Nickname" → Safe)	Score increases; feedback is shown
6	Complete all sorting items	Timer stops, score is finalized
7	Backend receives score and time via REST API	Submission is logged and acknowledged
8	Leaderboard updates to include current student's result	Updated score and rank are displayed

3.2 Case - 1: Level 2 Game: Password Security Challenge

3.2.1 Purpose

To verify the functionality of Level 2 (Password Security) game, including password creation mechanics, timer accuracy, scoring based on password strength, and leaderboard update after level completion.

3.2.2 Inputs

Input Type	Description
Game Level Selection	User selects **Password Security** level from the main game menu
Timer Trigger	Timer starts automatically when the level begins
Timer Completion Trigger	Level ends when password submitted or timer expires
Score Submission Trigger	Final score based on password strength and time sent to backend
Leaderboard Update Trigger	Leaderboard fetches and updates player score/time

3.2.3 Expected Outputs & Pass/Fail Criteria

Related Input	Expected Output	Pass/Fail Criteria
Game Level Selection	Game environment and UI for Level 2 load properly	**Pass** if level loads without errors; **Fail** if UI or environment fails to load
Timer Trigger	Timer starts at the beginning of the level	**Pass** if timer starts immediately; **Fail** if timer doesn't start or delayed
Timer Completion Trigger	Timer stops after password submission or timeout	**Pass** if timer stops correctly; **Fail** if timer continues running or stops incorrectly
Score Submission Trigger	Final score and time submitted via API	**Pass** if score/time sent correctly; **Fail** if missing or incorrect
Leaderboard Update Trigger	Leaderboard updates with latest player score/time	**Pass** if leaderboard updates correctly; **Fail** if not updated or incorrect

3.2.4 Test Procedure

Step No.	Test Action	Expected Result
1	Launch the game and select **Password Security** level	Level 2 loads; game environment initializes
2	Timer begins as game starts	Timer starts counting
3	Complete story line game	Timer stop triggers
4	Timer stops after game complete	Timer stops, total time is recorded
5	Backend receives score and time via REST API	Submission is logged and acknowledged
6	Leaderboard updates to include current student's result	Updated score and rank are displayed

3.3 Case - 1: Level 3 Game: Phishing Identification Challenge

3.3.1 Purpose

To validate the functionality of Level 3 (Phishing Identification Challenge), where players identify phishing attempts across different scenarios (email, SMS, deceptive phishing). This includes verifying player actions (spam, archive, delete, etc.), real-time feedback on correctness, timer accuracy, score calculation, and leaderboard updates.

3.3.2 Inputs

Input Type	Description
Game Level Selection	User selects **Phishing Identification Challenge** level from the main menu
Timer Trigger	Timer starts automatically when the level begins
Player Action – Scenario Selection	Player views and inspects phishing scenarios (Email, SMS, Deceptive phishing)
Player Action – Phishing Identification	Player chooses an action on each scenario (Spam, Archive, Delete, etc.)
Feedback Trigger	System provides immediate feedback (correct/incorrect + explanation) after each action
Timer Completion Trigger	Level ends when player finishes all scenarios or timer runs out
Score Submission Trigger	Final score based on correctness and time submitted to backend
Leaderboard Update Trigger	Leaderboard fetches and updates player score/time

3.3.3 Expected Outputs & Pass/Fail Criteria

Related Input	Expected Output	Pass/Fail Criteria
Game Level Selection	Game environment and UI for Level 3 load properly	**Pass** if UI and environment load correctly; **Fail** if any loading issues
Timer Trigger	Timer starts at the beginning of the level	**Pass** if timer starts immediately; **Fail** if timer doesn't start or delayed
Player Action – Scenario Selection	Phishing scenarios are presented clearly to the player	**Pass** if scenarios display properly; **Fail** if scenarios are missing or unclear
Player Action – Phishing Identification	Actions (spam, archive, delete, etc.) are selectable and work correctly	**Pass** if actions are selectable and trigger correctly; **Fail** if actions fail or are unresponsive
Feedback Trigger	Immediate feedback after each action indicating correctness with explanation	**Fail** due to missing/improper feedback (needs fix)
Timer Completion Trigger	Timer stops after all scenarios completed or timeout	**Pass** if timer stops correctly; **Fail** if timer continues or stops early
Score Submission Trigger	Final score and completion time submitted via API	**Pass** if data is submitted correctly; **Fail** if submission fails
Leaderboard Update Trigger	Leaderboard updates with the latest player score/time	**Fail** because leaderboard update does not function correctly

3.3.4 Test Procedure

Step No.	Test Action	Expected Result
1	Launch the game and select **Phishing Identification Challenge** level	Level 3 loads; game environment initializes
2	Timer begins automatically as game starts	Timer starts counting
3	Player inspects phishing scenarios: email, SMS, and deceptive phishing	All three types of scenarios are displayed clearly
4	Player selects actions (spam, archive, delete, etc.) on each scenario	Actions are selectable and register correctly
5	After each action, immediate feedback is shown indicating if the choice was correct or incorrect with explanation	Feedback matches player's choice and explains correctness
6	Player completes all phishing scenarios or timer expires	Timer stops, total time recorded
7	Backend receives final score and time via REST API	Submission is logged and acknowledged
8	Leaderboard updates to reflect current player’s score and rank	Updated score and rank are displayed

3.4 Case - 1: Timer System

3.4.1 Purpose

To verify that the timer system in all game levels starts, stops, and records elapsed time correctly, and that it integrates properly with scoring and leaderboard updates.

3.4.2 Inputs

Input Type	Description
Game Level Start	Trigger to start the timer automatically when a game level begins
Level End / Completion	Trigger to stop the timer when the player finishes the level
Game Exit/Abort	Player exits or aborts the game level before completion

3.4.3 Expected Outputs & Pass/Fail Criteria

Related Input	Expected Output	Pass/Fail Criteria
Game Level Start	Timer starts counting as soon as the level loads	**Pass** if timer starts immediately; **Fail** if delayed or no start
Level End / Completion	Timer stops when the player completes the level or when timer expires	**Pass** if timer stops correctly; **Fail** if it continues running or stops early
Game Exit/Abort	Timer stops and elapsed time is recorded up to the point of exit	**Pass** if elapsed time is accurately recorded; **Fail** if not recorded or incorrect

3.4.4 Test Procedure

Step No.	Test Action	Expected Result
1	Start any game level (Level 1, 2, or 3)	Timer starts counting immediately
2	Play through part of the level	Timer counts elapsed time continuously
3	Complete the level	Timer stops exactly when level ends
4	Check that the recorded elapsed time is correctly displayed or stored	Elapsed time matches real-world time spent
5	Exit the game level abruptly before completion	Timer stops and records elapsed time until exit
6	Submit score and time to backend	Backend receives accurate time data
7	Verify leaderboard shows player’s completion time accurately	Leaderboard displays correct time ranking

3.5 Case - 1: Scoring System

3.5.1 Purpose

To verify that the scoring system accurately calculates and updates scores based on player actions across all game levels, including correct and incorrect responses, penalties, and integration with timer and leaderboard.

3.5.2 Inputs

Input Type	Description
Correct Action	Player performs correct action (e.g., correct classification, strong password, correct phishing flag)
Incorrect Action	Player performs incorrect action (e.g., misclassification, weak password, wrong phishing flag)
Score Submission Trigger	End of level or player submits score

3.5.3 Expected Outputs & Pass/Fail Criteria

Related Input	Expected Output	Pass/Fail Criteria
Correct Action	Player awarded positive points according to predefined scoring rules	**Pass** if scoring is accurate for correct actions; **Fail** if points are missing or incorrect
Incorrect Action	Player receives zero points	**Pass** if no points awarded for incorrect actions; **Fail** if points incorrectly awarded
Score Submission Trigger	Final score submitted to backend service for leaderboard update	**Pass** if score is submitted correctly; **Fail** if submission fails or is delayed

3.5.4 Test Procedure

Step No.	Test Action	Expected Result
1	Start a game level and perform a series of correct actions	Score increases by correct amount for each action
2	Perform incorrect actions	Score decreases or no points awarded, penalties applied correctly
3	Continue playing with mixed correct and incorrect actions	Score updates cumulatively and correctly after each action
4	Complete the level and submit score	Final score calculated accurately including timer effects
5	Verify submitted score is recorded in backend	Backend receives correct final score
6	Check leaderboard reflects updated score	Leaderboard shows accurate player ranking based on score

3.6 Case - 1: Leaderboard System

3.6.1 Purpose

To verify that the leaderboard system correctly ranks players based on their scores and completion times, updates dynamically after each game session, and displays accurate, real-time standings to users and teachers.

3.6.2 Inputs

Input Type	Description
Player Score	Final score submitted after game level completion
Player Completion Time	Time taken by the player to complete the level
Score & Time Submission	Data sent from the game client to backend leaderboard service
Request to Fetch Leaderboard	Query from clients or teachers to display the current leaderboard data

3.6.3 Expected Outputs & Pass/Fail Criteria

Step No.	Test Action	Expected Result	Pass/Fail Criteria
1	Submit multiple player scores and completion times to leaderboard	Leaderboard updates with new data	**Pass** if leaderboard updates immediately; **Fail** if delayed or no update
2	Fetch leaderboard from game client or teacher dashboard	Leaderboard shows players ranked correctly by score and time	**Pass** if ranking is accurate; **Fail** if incorrect or incomplete data
3	Verify top player is the one with highest score; ties broken by time	Ranking order reflects scoring rules	**Pass** if ranking logic is correct; **Fail** if errors in sorting occur
4	Check that individual player can view their accurate rank	Player's rank corresponds with their score and time	**Pass** if player sees correct rank; **Fail** if rank is wrong or missing

3.6.4 Test Procedure

Step No.	Test Action	Expected Result
1	Submit multiple player scores and completion times to leaderboard	Leaderboard updates with new data
2	Fetch leaderboard from game client or teacher dashboard	Leaderboard shows players ranked correctly by score and time
3	Verify top player is the one with highest score; ties broken by time	Ranking order reflects scoring rules
4	Check that individual player can view their accurate rank	Player's rank corresponds with their score and time

3.7 Case - 1: Student Registration System

3.7.1 Purpose

To verify that students can register by entering their real name and a valid class code, ensuring they are correctly enrolled in their designated class. The system must reject invalid class codes and prevent unauthorized access to the game and dashboard.

3.7.2 Inputs

Input Type	Description
Real Name	Student enters their full real name
Class Code	Student inputs the class code provided by the teacher
Roblox ID	Unique Roblox identifier linked to the student

3.7.3 Expected Outputs & Pass/Fail Criteria

Related Input	Expected Output	Pass/Fail Criteria
Valid Class Code	Student is successfully registered, linked to Roblox ID, and enrolled in correct class	**Pass** if registration and enrollment succeed correctly; **Fail** if errors occur or class assignment is wrong
Invalid Class Code	Registration is rejected; student cannot enter the game or access dashboard	**Pass** if system blocks registration and access properly; **Fail** if registration allowed or no error shown
Missing or Empty Real Name or Class Code	Registration is rejected with appropriate validation message	**Pass** if validation message shown and registration blocked; **Fail** if no message or incomplete validation
Real Name (secure storage)	Real name is encrypted upon storage and decrypted only for	**Pass** if encryption is applied at storage and decrypted correctly for teachers; **Fail** if data is stored in plain text, not

	viewing by authorized teachers in the reporting dashboard	decrypted, or accessible by unauthorized users

3.7.4 Test Procedure

Step No.	Test Action	Expected Result
1	Student enters real name and a valid class code	Registration successful; student enrolled in designated class
2	Student enters real name and an invalid or incorrect class code	Registration rejected; error message displayed; no enrollment
3	Student leaves real name or class code empty and attempts to register	Registration rejected; prompt to fill required fields
4	Verify that registered student data (real name, Roblox ID, class enrollment) appears in teacher's dashboard	Data correctly displayed and linked in dashboard
5	Attempt to enter game after successful registration	Student gains access and gameplay starts
6	Attempt to enter game with failed registration	Access denied; student cannot proceed to game
7	Check database storage of real names after registration	Real names are stored in encrypted form, not plain text
8	Authorized teacher views reporting dashboard with registered student data	Real names are decrypted and visible only to authorized teachers
9	Unauthorized user attempts to access student data	Real names remain inaccessible; no decrypted data is shown

3.8 Case - 1: Lock/Unlock Levels

3.8.1 Purpose

To verify that teachers can successfully lock or unlock specific game levels/worlds via the teacher dashboard, controlling student access to those levels in the Roblox game.

3.8.2 Inputs

Input Type	Description
Teacher ID	Identifier of the logged-in teacher
Student ID	Identifier of the student whose level access is controlled
Level ID	Identifier of the game level to be locked/unlocked
Lock/Unlock Command	Action to lock (disable) or unlock (enable) the level

3.8.3 Expected Outputs & Pass/Fail Criteria

Related Input	Expected Output	Pass/Fail Criteria
Lock command	Specified level is locked for the selected student; student access denied in-game	**Pass** if level access is blocked as intended; **Fail** if student can still access locked level
Unlock command	Specified level is unlocked for the selected student; student access granted in-game	**Pass** if level access is granted properly; **Fail** if student is still blocked or access fails
Confirmation message	Teacher dashboard shows success confirmation after action	**Pass** if confirmation shown immediately; **Fail** if no confirmation or delayed message

3.8.4 Test Procedure

Step No.	Test Action	Expected Result
1	Teacher logs into dashboard and selects a student	Student profile loads correctly
2	Teacher locks a specific game level for the student	System confirms level locked; student cannot access level
3	Student tries to enter locked level in Roblox game	Access denied; appropriate message or redirection occurs
4	Teacher unlocks the same level for the student	System confirms level unlocked; student can access level
5	Student enters unlocked level successfully	Level loads and gameplay starts normally

3.9 Case - 1: Reassign Students to Levels

3.9.1 Purpose

To verify that the teacher dashboard can serve as a backup mechanism for student progression. Specifically, this test ensures that teachers can manually reassign students to the next game level if automatic level advancement in Roblox fails. Successful reassignment should grant access to the new level while revoking access to the previous one.

3.9.2 Inputs

Input Type	Description
Teacher ID	Identifier of the logged-in teacher
Student ID	Identifier of the student to be reassigned
New Level ID	Identifier of the new game level to assign to the student

3.9.3 Expected Outputs & Pass/Fail Criteria

Related Input	Expected Output	Pass/Fail Criteria
Valid Student ID and New Level ID	Student's game level is updated to the new level in the system and teacher dashboard	**Pass** if level updates correctly and reflects on dashboard; **Fail** if update fails or not reflected
Invalid Student ID or New Level ID	System shows an error message and does not update the student's level	**Pass** if error shown and no changes made; **Fail** if system allows invalid reassignment
Student accesses game after reassignment	Student can only access the newly assigned level; previous level is inaccessible	**Pass** if access restrictions apply correctly; **Fail** if old level remains accessible or new level blocked

3.9.4 Test Procedure

Step No.	Test Action	Expected Result
1	Teacher logs into dashboard and selects a student	Student profile loads correctly
2	Teacher selects a new level and reassigns the student	System confirms reassignment success
3	Verify that student's assigned level is updated in the dashboard	Dashboard shows new level assignment
4	Student logs into Roblox game and attempts to access the old level	Access denied for old level
5	Student accesses the newly assigned level	Level loads and gameplay starts normally
6	Teacher attempts reassignment with invalid student or level IDs	System displays error without crashing

3.10 Case - 1: Class Creation and Enrollment

3.10.1 Purpose

To verify that teachers can create new class sections and enroll students under specific grade levels and sections through the teacher dashboard, ensuring proper class setup and student association.

3.10.2 Inputs

Input Type	Description
Teacher ID	Identifier of the logged-in teacher
Grade Level	Grade level assigned to the class (e.g., Grade 5)
Section Name	Section or subgroup name (e.g., Section A)

3.10.3 Expected Outputs & Pass/Fail Criteria

Related Input	Expected Output	Pass/Fail Criteria
Valid Class Name, Grade Level, Section	New class is created successfully and appears on teacher dashboard	**Pass** if class creation succeeds and is visible; **Fail** if creation fails or not shown
Student IDs provided	Students are enrolled and linked correctly to the new class	**Pass** if students are enrolled properly; **Fail** if enrollment fails or students missing
Duplicate Class Name	System shows error or prompts about duplicate class name	**Fail** if duplicate allowed; **Pass** if duplicate prevented with proper message
Invalid inputs (e.g., empty fields)	Validation error shown, and class creation is blocked	**Pass** if system prevents creation and shows error; **Fail** if invalid data is accepted

3.10.4 Test Procedure

Step No.	Test Action	Expected Result
1	Teacher logs into dashboard and navigates to class creation	Class creation form loads correctly
2	Teacher enters class name, grade level, and section	Input fields accept valid data
3	Teacher submits the form to create class	Confirmation message shown; class appears on dashboard
4	Teacher attempts to create a class with missing or invalid data	Validation errors prevent creation
5	Teacher tries to create duplicate class name	System prompts error or prevents duplication

3.11 Case - 1: Microsoft Teams Authentication

3.11.1 Purpose

To verify that teachers can securely log into the teacher dashboard using Microsoft Teams OAuth authentication, ensuring authorized access control and proper session management.

3.11.2 Inputs

Input Type	Description
Teacher Microsoft Teams Email	Registered email associated with Microsoft Teams account
Password	Corresponding password for Microsoft Teams account
OAuth Token	Authentication token returned by Microsoft OAuth service

3.11.3 Expected Outputs & Pass/Fail Criteria

Related Input	Expected Output	Pass/Fail Criteria
Valid Microsoft Teams credentials	Successful login; teacher redirected to dashboard	**Pass** if login succeeds and dashboard loads correctly; **Fail** if login fails or no redirect
Invalid credentials	Login rejected; error message displayed	**Pass** if login is blocked and clear error message is shown; **Fail** if login is allowed or no message
Expired/Invalid OAuth token	Login rejected; user prompted to reauthenticate	**Pass** if system detects invalid token and prompts for reauthentication; **Fail** if access is granted without prompt

3.11.4 Test Procedure

Step No.	Test Action	Expected Result
1	Teacher clicks "Login with Microsoft Teams" button	Redirected to Microsoft OAuth login page
2	Teacher enters valid Microsoft Teams email and password	OAuth token issued and redirected back to dashboard
3	Teacher enters invalid credentials	Error message shown, login denied
4	Teacher tries to login with expired/invalid OAuth token	Prompted to login again

3.12 Case - 1: Real-Time Notification System

3.12.1 Purpose

To verify that the teacher dashboard:

1. Displays real-time notifications when a student completes a level or registers.
2. Allows teachers to delete notifications.
3. Ensures that deleted notifications are not removed from the database but are flagged appropriately for auditing.

3.12.2 Inputs

Input Type	Description
Student Level Completion	Student completes a level in Roblox game
Student Registration	Student successfully registers using real name and valid class code
Teacher Deletes Notification	Teacher clicks "Delete" or "Clear All" for one or multiple notifications

3.12.3 Expected Outputs & Pass/Fail Criteria

Related Input	Expected Output	Pass/Fail Criteria
Student completes a level	Real-time notification appears on teacher dashboard with student name and level info	Pass if notification appears promptly with correct student name and level; Fail if missing or incorrect
Student registers	Real-time notification appears on teacher dashboard showing student's name & class	Pass if notification appears promptly with correct student name and class; Fail if missing or incorrect
Teacher deletes notification	Notification is removed from UI and marked as "deleted" (flagged) in database	Pass if notification is removed from UI and flagged as deleted in database; Fail if either condition not met
Notification timeout or refresh	Notifications remain visible unless explicitly deleted	Pass if notifications persist after timeout or refresh; Fail if they disappear without deletion

3.12.4 Test Procedure

Step No.	Test Action	Expected Result
1	Student registers using real name and valid class code	Real-time registration notification appears on teacher dashboard instantly
2	Student completes a game level	Level completion notification appears on dashboard with correct level ID and time
3	Teacher views the list of notifications	All recent registration and completion events are shown
4	Teacher clicks “Delete” on a single notification	Notification disappears from dashboard UI; backend record is flagged as deleted
5	Teacher clicks “Clear All” to remove all notifications	All notifications are removed from UI; backend records are flagged
6	Check backend logs or DB after deletion	Records are marked as deleted, not physically removed
7	Simulate 5+ notifications in sequence	Dashboard queues and displays all in real time without performance lag
8	Reload dashboard after several notifications	Undeleted notifications persist; deleted ones do not reappear

3.13 Case - 1: Online/Offline Status Tracking

3.13.1 Purpose

To verify that the teacher dashboard accurately tracks and displays the real-time **online** or **offline** status of each student currently logged into the Roblox game. This ensures that teachers can monitor student activity during gameplay sessions.

3.13.2 Inputs

Input Type	**Description**
Student Login Event	Student logs into the Roblox CyberKids game
Student Logout Event	Student logs out, disconnects, or exits the game
Status Poll Interval	System sends periodic or real-time WebSocket updates to the teacher dashboard

3.13.3 Expected Outputs & Pass/Fail Criteria

Related Input	**Expected Output**	**Pass/Fail Criteria**
Student logs in	Dashboard shows student as **"Online"** with green indicator	Pass if dashboard updates student status to “Online” with green indicator immediately; Fail if not
Student logs out or disconnects	Dashboard updates student status to **"Offline"** with red/gray indicator	Pass if dashboard updates status to “Offline” with red/gray indicator promptly; Fail if not
Multiple students active	Dashboard lists accurate status for all logged-in and offline students	Pass if all students’ statuses are correctly displayed as online or offline; Fail if any mismatch

3.13.4 Test Procedure

Step No.	Test Action	Expected Result
1	Student logs into Roblox game	Teacher dashboard shows student status as "Online"
2	Student logs out or closes the Roblox game	Teacher dashboard updates student status to "Offline"
3	Multiple students log in simultaneously	Dashboard shows all active students as "Online"
4	Disconnect a student abruptly (e.g., close window, lose internet)	System detects disconnection and updates status to "Offline"
5	Reconnect a student	Status changes back to "Online" in real-time
6	Refresh the teacher dashboard	Current online/offline statuses are accurately retained and displayed

Appendix (Test Logs)

A.1 Log for Test 3.1 - Level 1 Game: Information Classification Sorting Challenge

Step	Action Performed	Expected Result	Actual Result	Status
1	Teacher enters 6 custom information items via the dashboard	Items are saved to backend database	Items saved successfully	Pass
2	Student starts Level 1 in Roblox	Timer begins automatically	Timer started at 00:00 upon level load	Pass
3	Student sorts item “Nickname” to **Safe to Share** bin	+Points awarded for correct classification	Score increased by 10 points	Pass
4	Student sorts item “Home Address” to **Safe to Share** bin	No points awarded for incorrect classification	No points added; feedback popup shown	Pass
5	Student completes sorting all 6 items	Timer stops; score and time are recorded	Timer stopped at 00:45, score: 40	Pass
6	System sends score and time to backend	Data is transmitted to backend via REST API	Request sent to /submitScore; response: 200 OK	Pass
7	Leaderboard fetches and updates with new entry	Student’s score and time appear at correct rank	Leaderboard did **not** update; no new entry shown	**Fail**

A.1 Log for Test 3.2 - Level 2 Game: Password Security Challenge

Step	Action Performed	Expected Result	Actual Result	Status
1	User selects Password Security from main game menu	Game environment for Level 2 loads	Level 2 UI and environment loaded successfully	Pass
2	Level begins; user is prompted to collect and combine password fragments	Timer starts automatically	Timer initialized at 00:00	Pass
3	User creates password using collected items (e.g., "K9!Planet_2025")	System scores password based on strength and submits	Password evaluated as “Strong” (Score: 90)	Pass
4	User clicks Submit	Timer stops and final score is submitted via API	Timer stopped at 00:37; API call made to /submitScore – 200 OK	Pass
5	System attempts to update leaderboard	Score and time should appear on leaderboard	Leaderboard did not reflect updated player score	Fail

A.1 Log for Test 3.3 - Level 3 Game: Phishing Identification Challenge

Step	Action Performed	Expected Result	Actual Result	Status
1	User selects **Phishing Identification Challenge** from main menu	Game environment and UI for Level 3 load successfully	Level 3 UI and phishing scenarios loaded properly	Pass
2	Level starts	Timer begins automatically	Timer started at 00:00	Pass
3	Player inspects Email Phishing scenario	Scenario is presented clearly with readable content	Email phishing sample displayed correctly	Pass
4	Player marks a phishing email as **Spam**	Action is accepted, triggers evaluation	Action accepted; no feedback shown	**Fail**
5	Player moves on to SMS Phishing and Deceptive Phishing scenarios	Scenarios are displayed and navigable	All scenarios displayed correctly	Pass

A.1 Log for Test 3.4 - Timer System

Step	Action Performed	Expected Result	Actual Result	Status
1	Game level starts, triggering timer start	Timer begins counting automatically	Timer started at level load	Pass
2	Player completes level, triggering timer stop	Timer stops and elapsed time is recorded	Timer stopped and time recorded	Pass
3	Player aborts/exits level mid-game	Timer stops and elapsed time recorded up to exit point	Timer stopped and recorded correctly on exit	Pass

A.1 Log for Test 3.5 - Scoring System

Step	Action Performed	Expected Result	Actual Result	Status
1	Player performs a correct action	Player awarded positive points according to scoring rules	Points awarded correctly	Pass
2	Player performs an incorrect action	Player receives zero points	No points awarded	Pass
3	Player finishes level and triggers score submission	Final score submitted to backend successfully	Score submitted via API	Pass

A.1 Log for Test 3.6 - Leaderboard System

Step	Action Performed	Expected Result	Actual Result	Status
1	Player submits final score and completion time	Leaderboard ranks players by score (primary) and completion time (secondary)	Players ranked correctly based on score and time	Pass
2	Leaderboard receives new score/time submission	Leaderboard updates immediately to reflect new player data	Leaderboard updates successfully	Pass
3	Client or teacher requests to fetch leaderboard data	Leaderboard data is returned with correct top players and scores	Data returned but contains outdated or incomplete information	Fail
4	Player views leaderboard on UI	Player sees their accurate rank and score displayed	Player rank or score not displayed correctly or not updated	Fail

A.1 Log for Test 3.7 - Student Registration System

Step	Action Performed	Expected Result	Actual Result	Status
1	Student enters valid real name and valid class code	Student is successfully registered, linked with Roblox ID, and enrolled in class	Registration successful; student enrolled	Pass
2	Student enters invalid class code	Registration rejected; student prevented from entering game and dashboard	Registration rejected as expected	Pass
3	Student leaves real name or class code empty	Registration rejected with validation message	Appropriate validation error shown	Pass
4	System stores student data in backend	Student data saved and linked correctly to Roblox ID and class	Data saved correctly	Pass
5	Student attempts to enter game after registration	Access granted only if registration succeeded	Access granted/restricted correctly	Pass
6	The system shall encrypt and securely store all submitted real names ensuring that personal data is accessible only to authorized teachers through the reporting dashboard.	Only authorized teachers can view the decrypted real names through the reporting dashboard, while unauthorized users cannot access or view the information	Real names are stored in plain text. No encryption or decryption is applied. Data is visible without restriction.	Fail

A.1 Log for Test 3.8 - Lock/Unlock Levels

Step	Action Performed	Expected Result	Actual Result	Status
1	Teacher sends Lock command for a specific level to student	Level is locked for that student; student cannot access level in Roblox	Level locked and access denied as expected	Pass
2	Teacher sends Unlock command for a specific level to student	Level is unlocked for that student; student can access level in Roblox	Level unlocked and access granted as expected	Pass
3	Teacher dashboard shows confirmation message after lock/unlock action	Dashboard displays success message confirming action	Confirmation message displayed correctly	Pass
4	System updates backend database with lock/unlock status	Database reflects current locked/unlocked state of levels	Database updated correctly	Pass
5	Student attempts to access locked level	Access denied; appropriate message shown	Access blocked as expected	Pass

A.1 Log for Test 3.9 - Reassign Students to Levels

Step	Action Performed	Expected Result	Actual Result	Status
1	Teacher inputs valid Student ID and New Level ID in dashboard	Student's level is updated in backend and visible in teacher dashboard	Level updated correctly	Pass
2	Teacher inputs invalid Student ID or New Level ID	System displays error message; no level change occurs	Error message displayed correctly	Pass
3	Student attempts to access previous level after reassignment	Access denied to previous level; access granted only to newly assigned level	Access permissions updated correctly	Pass
4	Dashboard reflects the updated level for reassigned student	Teacher dashboard shows the student's current assigned level	Dashboard data updated as expected	Pass
5	Backend database reflects updated level assignment	Database shows new level for student and revoked access to old level	Database updated successfully	Pass

A.1 Log for Test 3.10 - Class Creation and Enrollment

Step	Action Performed	Expected Result	Actual Result	Status
1	Teacher inputs valid Grade Level and Section Name	New class is created and appears on teacher dashboard	Class created successfully	Pass
2	Teacher enrolls student IDs into the newly created class	Students are linked to the new class and visible in dashboard	Students enrolled correctly	Pass
3	Teacher tries to create a class with a duplicate Grade & Section	System shows error or warning about duplicate class	Can be duplicated	Fail
4	Teacher submits empty or invalid fields during class creation	System prevents creation and shows validation errors	Validation error triggered correctly	Pass
5	Dashboard updates to reflect the newly created classes and enrollments	Dashboard lists new classes and enrolled students accurately	Dashboard updated correctly	Pass

A.1 Log for Test 3.11 - Microsoft Teams Authentication

Step	Action Performed	Expected Result	Actual Result	Status
1	Teacher enters valid Teams email and password	Successful login; redirected to dashboard	Login successful	Pass
2	Teacher enters incorrect password	Login rejected; error message displayed	Error message shown	Pass
3	OAuth token expired or invalid	Login rejected; reauthentication prompt shown	Reauthentication prompted	Pass
4	Network error during authentication	Proper error handling and retry mechanism	Error handled gracefully	Pass

A.1 Log for Test 3.12 - Real-Time Notification System

Step	Action Performed	Expected Result	Actual Result	Status
1	Student completes a level in Roblox game	Real-time notification appears on teacher dashboard	Notification did not appear immediately	Fail
2	Student registers with valid name & class code	Real-time notification appears on teacher dashboard	Notification appeared instantly	Pass
3	Teacher deletes a notification	Notification removed from UI and flagged as deleted in database	Notification removed and flagged in database	Pass
4	Wait for notification timeout or refresh	Notifications remain visible unless explicitly deleted	Notifications disappeared automatically	Fail

A.1 Log for Test 3.13 - Online/Offline Status Tracking

Step	Action Performed	Expected Result	Actual Result	Status
1	Student logs into the game	Dashboard shows student as "Online" with green indicator	Dashboard shows student as "Online" with green indicator	Pass
2	Student logs out/disconnects	Dashboard updates status to "Offline" with red/gray indicator	Dashboard updates status to "Offline" with red/gray indicator	Pass
3	Multiple students log in/log out	Dashboard accurately reflects each student's online/offline status	Dashboard accurately reflects all students' statuses	Pass
4	Status updates via WebSocket/poll	Real-time status updates sent and reflected on dashboard	Real-time status updates sent and reflected on dashboard	Pass

A.2 Test Results

Test Case ID	Description	Tester	Date	Result	Remarks
3.1	Info Classification Sorting Challenge	Ashley Vequiso	09/01/2025	Fail	Leaderboard not updating after score submit
3.2	Password Security Challenge	Jesson Cultura	09/01/2025	Fail	Leaderboard not updating after score submit
3.3	Phishing Identification Challenge	Emmanuel Dedumo	09/01/2025	Fail	Feedback not triggered; leaderboard failed to update
3.4	Timer System	John Kenneth Baguio	09/02/2025	Pass	Timer system working as expected
3.5	Scoring System	Jesson Cultura	09/02/2025	Pass	Scoring system working as expected
3.6	Leaderboard System	Ashley Vequiso	09/02/2025	Fail	Issues with fetching leaderboard data and player rank display
3.7	Student Registration System	Jesson Cultura	09/02/2025	Fail	Real names are stored in plain text without encryption/decryption
3.8	Lock/Unlock Levels	Emmanuel Dedumo	09/02/2025	Pass	Teacher dashboard controls working perfectly
3.9	Reassign Students to Levels	John Kenneth Baguio	09/02/2025	Pass	Manual reassignment works smoothly
3.10	Class Creation and Enrollment	Jesson Cultura	09/03/2025	Fail	Section and Grade name can still be duplicated

3.11	Microsoft Teams Authentication	Ashley Vequiso	09/03/2025	Pass	All authentication scenarios tested successfully
3.12	Real-Time Notification System	Jesson Cultura	09/04/2025	Fail	Fails to display for level completions and have issues with notification persistence on timeout or refresh.
3.13	Online/Offline Status Tracking	John Kenneth Baguio	09/04/2025	Pass	The system accurately tracks and displays online/offline statuses in real-time on the teacher dashboard.

A.3 Incident Report

Incident ID	Related Test Case ID	Description	Steps to Reproduce	Expected Result	Severity	Status	Reported By	Date
INC-01	3.1, 3.2	Leaderboard does not update after score submission	1. Complete Level 1 or 2 2. Submit score3. Refresh leaderboard	Leaderboard shows new score and rank	High	Open	Ashley Vequiso Jesson Cultura	09/01/2025
INC-02	3.3	Feedback not triggered when flagging phishing email	1. Start Level 3 2. Flag suspicious email as spam	Feedback should appear (correct/incorrect)	High	Open	Emmanuel Dedumo	09/01/2025
INC-03	3.6	Leaderboard fetch returns outdated/incomplete data	1. Submit score 2. Teacher fetches leaderboard	Latest scores and ranks displayed	High	Open	Ashley Vequiso	09/02/2025
INC-04	3.12	Real-time notifications not showing for level completions	1. Complete a game level 2. Check teacher dashboard	Notification appears instantly (<2s)	Medium	Open	Jesson Cultura	09/04/2025
INC-05	3.12	Notifications disappear automatically instead of persisting	1. Wait after notification received 2. Refresh dashboard	Notifications remain until cleared	Medium	Open	Jesson Cultura	09/04/2025

INC-106	3.10	Class name, grade, and section can be duplicated without error	1. Go to class registration 2. Enter an existing class name, grade, and section that already exists in the system 3. Submit the class registration	System rejects the duplicate entry and displays an error message (e.g., *“Class already exists”*)	Medium	Open	Jesson Cultura	09/03/25
INC-107	3.7	Real names are stored in plain text without encryption/decryption	1. Register a student with real name, valid class code, and Roblox ID 2. Check database storage for the registered student’s real name 3. Log in as authorized teacher and view reporting dashboard	Real names are stored in encrypted form in the database and only authorized teachers can view decrypted real names in the dashboard	High	Open	Jesson Cultura	09/02/25