Test Plan

Group 4 - CERP Target Tracking

Elvis Kam (cqst66)
Millie Crawford (dsvg51)
Jasmeet Kaur Jasvinder Singh (fvtx23)
Hongyu Bu (hpgm94)
Rachel Dela Ysla (lwcl21)
Tirenioluwa Oladimeji (sxgm48)

06/02/24

Table of Contents

Section 1

Introduction

This document outlines the test plan for the development and implementation of the CERP Target Tracking system for Durham City Council.

Our aim is to develop a web application that will efficiently manage and assess the progress of their climate initiatives, replacing their current, inefficient Excel-based system. The new system strives to facilitate efficient progress tracking, through a user-friendly interface, optimising the council's resource management towards their net zero target whilst providing stakeholders with transparent visibility into project progress.

This test plan provides a comprehensive framework of the testing scope, strategy, and outcomes, ensuring our project strongly aligns with our client's requirements. We have planned to conduct a wide range of rigorous tests, which will confirm the strengths of our application, as well as highlight areas of opportunities to work on.

Document Structure Overview:

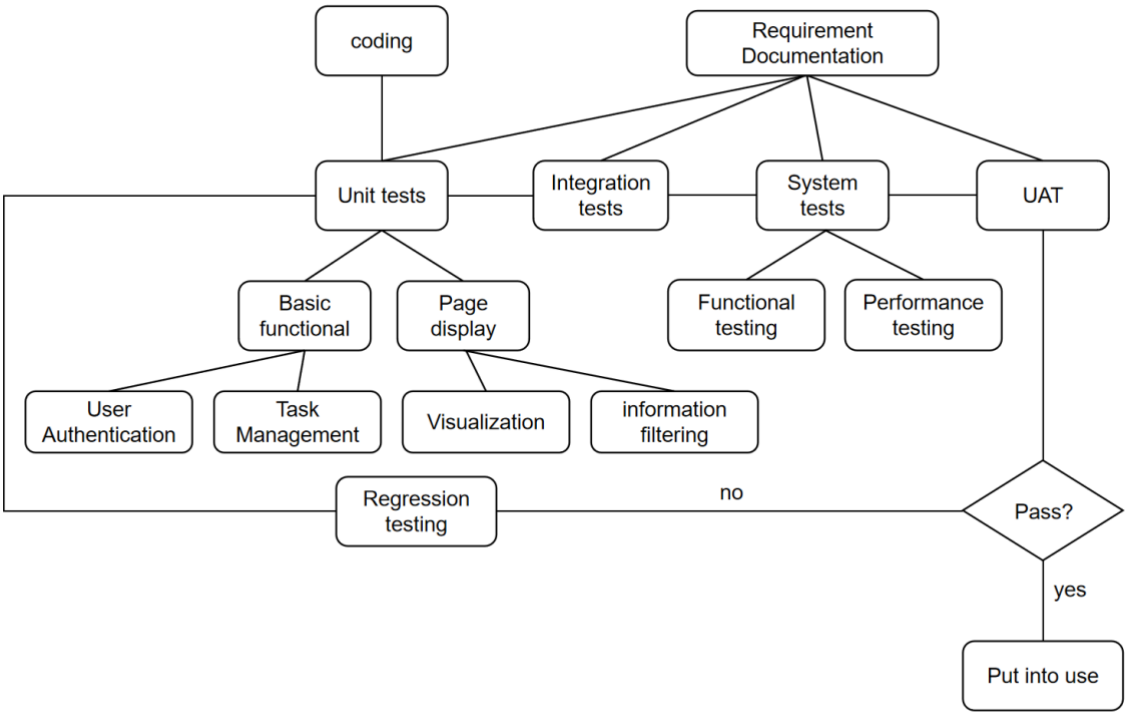
Our test plan is divided into three parts:

- **Main Items:**
  We have listed all the unit tests, integration tests, system tests, and user acceptance tests, and described our testing process and key focus areas.
- **Test Cases:**
  The test cases section samples from unit tests, integration tests, system tests, and user acceptance tests. It employs a standardized set of test procedures to evaluate the samples, and a test result form has been created to record the outcomes.
- **Testing Context:**
  This section primarily introduces the setup of the testing environment, the testing tools used, and the classification of fault severity levels.

Why Were Tests Chosen?

Within our tests we aim to cover all aspects of our web application. This allows us to ensure that the system functionality is correct, and the system meets the user requirements specified at the start of our project. The unit tests chosen will cover many different aspects from our system, ensuring that each area of the web application works as intended. The system tests again also cover a variety of areas, decreasing the chance of errors. Though our test cases could have been more comprehensive, this is a condensed list of what we plan to test, which will still cover all aspects of the entire system.

Tests



Overview

We employ a multi-layered, progressively in-depth testing strategy: unit testing → integration testing → system testing → user acceptance testing, as illustrated in the diagram above. This testing approach meets user requirements (as specified in the requirement documents) while enabling independent development, testing, and maintenance of individual functionalities, thereby reducing the risks encountered during integration and system testing and ensuring the stability and reliability of the system at final release.

After executing tests, we perform regression testing, which not only focuses on the areas where key functional modules have previously encountered issues but also ensures that each code modification does not introduce new defects. This approach covers all aspects, allowing for the timely detection and correction of potential regression errors.

In the table below, we have listed the contents covered by four types of tests, selecting four to five sample test cases from each category, totalling 17 test cases. Since our project is still in the development stage, we can only predict and display a portion of the actual results.

Currently, all tests used in the project are conducted as black-box tests. As we have only set up a simple front-end and written functions for some units, we can only obtain test results for the completed unit components, and some integration tests may not yield accurate results.

Unit Testing

In the software development process, our system widely adopts small components with clear functions and single responsibilities, while fully incorporating user requirements. The design of these components follows a well-defined division of labour: some components focus on implementing core functionalities such as task management and user authentication, while others are dedicated to visual display and information filtering. During the unit testing phase, validating the functional components is particularly critical, as they directly determine whether the system can accurately and efficiently execute the intended business logic. In contrast, visual components—primarily focused on interface presentation and enhancing user experience—cannot compensate for the negative impact on system stability and business processes if the functional components fail. Therefore, rigorous unit testing of the functional components to ensure their correct operation is far more important than merely focusing on the visual components.

Test ID	Description	Brief Test Oracle
unit-test_00	Router endpoint HTTP	Router endpoint returns correct HTTP status
unit-test_01	Sign up button	Click the ‘Sign up’ button to jump to the expected page
unit-test_02	Target Assignment and delete button	Specific elements should be displayed when manager clicks on a target, delete and assign target buttons
unit-test_03	Log out functionality	Close the current page and route to the login page

System Testing

System testing requires comprehensive testing of the entire software to ensure that all functionalities are complete and operate normally once all units have been integrated, while also taking into account the system's security, compatibility, and performance. Functional testing includes automated reminder functionality, access permission control, exception handling, and data processing functions, whereas performance testing involves response time and load testing.

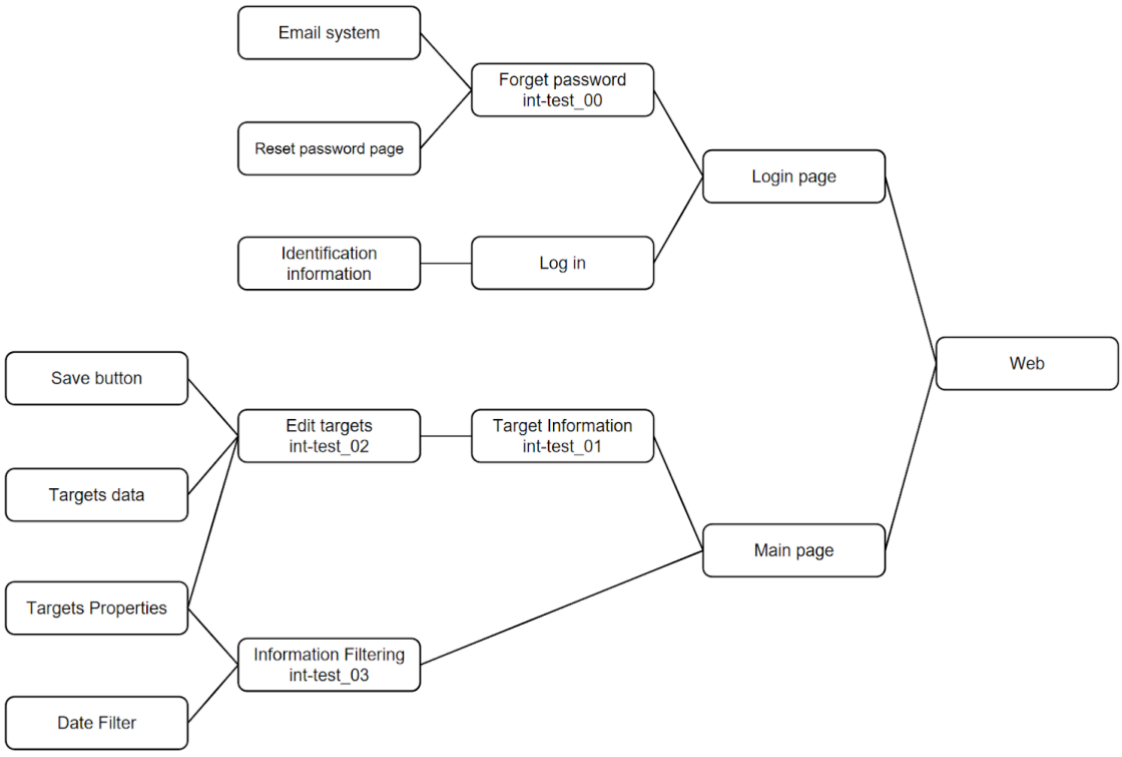
Test ID	Description	Functional/Non-functional	Brief Test Oracle
sys-test_00	Permission Management	Functional	Whether the system correctly determines whether a user is a manager or not; when logging in with a manager account, the system should accurately identify and grant access to the appropriate manager-only dashboards; for employee accounts, only access to the user's standard dashboards should be provided.
sys-test_01	Target information retrieval and display	Functional	The main interface should contain the target file, when the target ID is sent to the backend, the system will read and return the details of the specified target ID, then this data will be sent back to the frontend and displayed on the main interface
sys-test_02	Automated reminders	Functional	Get staff target information Send message reminder
sys-test_03	Profile information	Functional	Record username information at login and summarise them on the profile page

sys-test_04	Concurrency control	Non-Functional	Maintain data integrity and consistency when multiple people edit files

Integration Testing

After ensuring that the test results of all units meet the expected requirements, multiple unit modules are combined to verify whether they can complete the predetermined business process when working together and detect possible problems.
Methodology selection:
In our projects, the login system usually consists of several relatively independent modules, such as user interface, authentication logic, database interaction and error handling. It is possible to integrate the system modules in batches through incremental integration testing, integrating and testing a portion of the modules at a time, and then continuing the integration of the validated modules with other modules.

Test ID	Description	Brief Test Oracle
int-test_00	Forgotten password	Accept New Password Update the password data stored in the database
int-test_01	Target Information	Ensure that the dashboard returns correct and complete target information
int-test_02	Edit targets	Clicking the Edit button displays the stored target data, allowing the user to change and save new target data
int-test_03	Information Filtering	Find a list of targets that were active or created within the selected time range and display them in the main interface



User Acceptance Testing

User Acceptance Testing (UAT), as the final testing phase, primarily aims to verify whether the system fully meets the business requirements and user expectations. The testing focuses on the correctness of functionality, with an emphasis on user experience and interface friendliness. This phase is based on scenario testing derived from user stories and is ultimately executed by users who provide feedback.

Test ID	Description	Brief Test Oracle
ua-test_00	Responsive web	Automatically adjust the layout of your web app when running on a variety of devices and screen sizes
ua-test_01	Project progress visualisation	There is a progress bar above all tasks to view real-time task progress
ua-test_02	My Targets	The My Targets should only show projects that are relevant to me
ua-test_03	Screening Effectiveness Acceptance	The filtering function is intuitive and easy to use. All users can independently search for the desired target when using the filtering function.

Section 2

Test Cases

Unit Testing

Test 1

Test Case ID	**unit-test_00**
Description of test	Verify that router endpoints respond with correct HTTP code for each case.
Related requirement document details	N/A
Prerequisites for test	- Server is running - Server is reachable - Endpoints are defined and properly configured
Test procedure	1. Send requests to all router endpoints with appropriate methods (e.g. GET, POST, PUT, DELETE) 2. Verify that the response contains a valid HTTP status code 3. Compare the received status code against the expected status code for each endpoint 4. Log any mismatches between expected and actual results
Test material used	- Computer/mobile devices connected to the internet - Web browser (e.g. Google Chrome) - Test.js containing automated test cases for endpoints

Equivalence Classes	**Normal (Valid Inputs)**	Valid endpoints
	Boundary (Edge Cases)	Min/max parameters given for request endpoints
	Erroneous (Invalid Inputs)	Wrong endpoints
Expected result (test oracle)	The test **passes** if: each endpoint responds with the correct HTTP status code: • 200 for successful GET requests • 404 for not found • 401 for unauthorised access The test **fails** if: each endpoint responds with the incorrect or no HTTP status code.	
Comments	Ensure all required headers and authentication tokens are provided for secured endpoints.	
Created by	RDY	
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18	

Test 2

Test Case ID	**unit-test_01**	
Description of test	Verify that when the “Sign Up” button is clicked on the login page, the system correctly redirects the user to the sign-up page under different conditions.	
Related requirement document details	**Login system (Priority: Must have)** *“As a user, I want to be able to create an account so that I can securely log in to the system that is tailored towards me.”*	
Prerequisites for test	- Access to the functional web-application - The user is on the login page - JavaScript is enabled in the browser - The “Sign Up” button is visible and clickable	
Test procedure	1. User loads the web-application 2. Given that the user does not have an account, click the “Sign Up” button 3. Verify that the page changes to the sign-up page and the URL changes accordingly	
Test material used	- Functional web-application - Computer/mobile devices connected to the internet - Web browser (e.g. Google Chrome)	
Equivalence Classes	**Normal (Valid Input)**	Clicking the “Sign Up” button with JavaScript enabled E.g.: User clicks “Sign Up” and the page successfully redirects

	Boundary (Edge Cases)	Clicking “Sign Up” multiple times rapidly E.g.: The page should not reload multiple times and only redirects once
	Erroneous (Invalid Input)	1. Clicking the “Sign Up” button with JavaScript disabled E.g.: User clicks the “Sign Up” button, but the page does not redirect. An error message is displayed 2. Clicking the “Sign Up” button while the network is disconnected E.g.: User clicks but gets a network error message
Expected result (test oracle)	The test **passes** if: • The page changes from the Login page to the Sign-up page. • The URL updates correctly • The Sign-Up form appears with no broken elements The test **fails** if: • The page does not change from the Login page to the Sign-up page. • The wrong page is displayed instead of the Sign-up page. • The page redirects multiple times instead of once • The page redirects to an external malicious site • The page is blocked due to browser settings	
Comments	Test on multiple browsers (Chrome, Firefox, Edge, Safari) Check for accessibility (keyboard navigation, screen readers)	
Created by	MC	
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18	

Test 3

Test Case ID	**unit-test_02**
Description of test	Verify that when a manager clicks on a target, the delete and assign buttons appear, while staff members do not see these buttons.
Related requirement document details	**Ability to edit goals (Priority: Must have)** *“As a staff member I want to have the ability to edit goals because I need to keep track of the targets I have been set.”* **Permission levels (Priority: Must have)** *“As a manager, I want permission levels on the project so that my team cannot view other teams' projects in order to improve data security issues.”*
Prerequisites for test	- Access to the functional web-application - Valid user account - Target table populated with targets

Test procedure	1. Log in to the system as a manager 2. Under the “All Targets” section click on a target. 3. Verify that the target page is loaded up and at the bottom of the page, there is a delete button and an assign button 4. Log out and log in as a staff member 5. Under the “All Targets” section click on a target 6. Verify that the delete and assign buttons are not visible	
Test material used	- Functional web-application - Computer/mobile devices connected to the internet - Web browser (e.g. Google Chrome)	
Equivalence Classes	**Normal (Valid Input)**	Manager account E.g.: Manager clicks on a target, the delete and assign button appears
	Boundary (Edge Cases)	Manager with a large number of targets E.g.: Manager logs in with more than 100 targets. User interface does not crash, and buttons still appear correctly
	Erroneous (Invalid Input)	Staff account E.g.: Staff member logs in and clicks a target. The delete and assign buttons should not appear.
Expected result (test oracle)	The test **passes** if: • Managers can see the delete and assign buttons when selecting a target • Staff members do not see the delete and assign buttons • The UI remains responsive even when handling a large number of targets The test **fails** if: • The delete and assign buttons are missing when a manager selects a target • Staff members can see the delete and assign buttons • The UI crashes or hangs when many targets exist	
Comments	Test is conducted across multiple browsers	
Created by	MC	
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18	

Test 4

Test Case ID	**unit-test_03**
Description of test	Verify that when the user clicks on the ‘Log out’ button, the system logs the user out, and the login page shows**.**

Related requirement document details	**Login system (Priority: Must have)** *"As a user, I want to be able to create an account so that I can securely log in to the system that is tailored towards me."*	
Prerequisites for test	- Access to the functional web-application - User account	
Test procedure	1. Log in to the system as a user (staff or manager) 2. Click the profile icon in the top right corner 3. Click the 'Logout' button on the profile page 4. Verify that the Login page shows.	
Test material used	- Functional web-application - Computer/mobile devices connected to the internet - Web browser (e.g. Google Chrome)	
Equivalence Classes	**Normal (Valid Input)**	User clicks the Log out button E.g.: User logs out and is redirected to login page
	Boundary (Edge Cases)	User clicks Log out multiple times rapidly E.g.: The system should not crash, and user is logged out only once
	Erroneous (Invalid Input)	1. User manually enters dashboard URL after logging out E.g.: The dashboard page is not loaded, and the system redirects back to login page 2. User clicks back on the browser after logging out E.g.: The dashboard is not loaded, and the system redirects back to the login page
Expected result (test oracle)	The test **passes** if: • The user is relocated to the Login page, logging the user out of the account. • Using the browser's back button does not restore the previous session after logging out • Manually entering the dashboard URL redirects the user back to the Login page • The system does not crash due to repeated clicks on the Log out button The test **fails** if: • The user is not redirected to the Login page after clicking Log out, i.e. if it stays on the profile page, or the user is redirected to the dashboard. • Clicking the back button restores the session, allowing access to the dashboard • Manually entering the dashboard URL after logging out loads the dashboard page • The session remains active, and the user is not properly logged out • Unexpected error messages appear, i.e. "Logout Failed" or "Session Timeout")	
Comments	- Validate session termination by checking cookies and authentication tokens after logout	

	- Clear cache before each test to rule out browser caching behaviour affecting logout
Created by	MC
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18

System Testing

Test 5

Test Case ID	**sys-test_00**	
Description of test	Verify role-based dashboard access, ensuring the system correctly identifies the user as a manager or staff member upon login and grants access to the appropriate dashboards	
Related requirement document details	**Login system (Priority: Must have)** *"As a user, I want to be able to create an account so that I can securely log in to the system that is tailored towards me."* **Permission levels (Priority: Must have)** *"As a manager, I want permission levels on the project so that my team cannot view other teams' projects in order to improve data security issues."*	
Prerequisites for test	- Access to the functional web-application - User accounts (Include at least one manager account and one staff member account) - User table	
Test procedure	1. Login to the system using manager credentials 2. Verify that the system directs the user to the managers dashboard 3. Login to the system using staff credentials 4. Verify that the system directs the user to the staff dashboard 5. Test incorrect inputs (such as wrong passwords or usernames) to confirm that the system handles errors properly and displays the correct error messages. 6. Test permission boundaries, such as attempting to access the manager's page with a staff member account, to ensure the system denies access. 7. Enter the URL to the manager's page and check if it redirects to the 403-error page, verifying that the displayed message is "Access Denied." 8. Test various incorrect URL formats (including illegal characters, overly long URLs, and structurally incorrect path entries) to see if the system redirects to the 404-error page and displays the message "Page Not Found."	
Test material used	- Functional web-application - User accounts for staff member and manager - Staff dashboard & managers dashboard - Login system supports role recognition and page redirection - Computer/mobile devices connected to the internet - Web browser (e.g. Google Chrome)	
Equivalence Classes	**Normal (Valid Input)**	Manager/ Staff log in successfully using correct credentials

		E.g.: Users are redirected to their respective dashboards
	Boundary (Edge Cases)	Multiple incorrect logins causing account lockout E.g.: After five failed attempts, the account is locked
	Erroneous (Invalid Input)	Incorrect credentials or invalid URL used E.g.: User types in wrong password, an error message is shown. Using a non-existent URL redirect to “404 Page Not Found”
Expected result (test oracle)	The test **passes** if: • Each role is correctly directed to their respective dedicated pages after login, and the pages will display user information and functional options that match the expected permissions of the role. • Incorrect login attempts are properly recognised, and the system provides appropriate error messages. Access is denied to restricted areas • Invalid URLs are handled properly • Account lockout mechanism works after repeated failed attempts The test **fails** if: • Users can access unauthorised areas (E.g., staff can view manager dashboard) • The system does not display an error message for incorrect credentials • The system crashes or fails to load after incorrect login attempts • The account is not locked after multiple failed login attempts	
Comments	Test is conducted on different browsers Verify authentication tokens expire correctly after logout	
Created by	HB	
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18	

Test 6

Test Case ID	**sys-test_01**
Description of test	Verify that when a target is clicked, the frontend sends the correct targetID to the backend and retrieves the expected details for display.
Related requirement document details	**Clear and intuitive user interface (Priority: Must have)** *"As a user, I want the system design to be clear and intuitive so that it is easy to use and allows me to focus my full attention on my work."*
Prerequisites for test	- Access to the functional web-application - User is logged into their account - Targets table containing multiple targets and their details
Test procedure	1. Identify a target card on the dashboard. 2. Click on the target card.

	3. Verify that the system responds by displaying the correct target details. 4. Ensure the details match the expected content in the database or system configuration. 5. Validate responsiveness and clarity of the interface displaying the details.	
Test material used	- Functional web application - Dashboard and interactive target card - Developer tools (e.g. Postman for API validation) - Computer/mobile devices connected to the internet - Web browser (e.g. Google Chrome)	
Equivalence Classes	**Normal (Valid Input)**	Click on an existing target E.g.: System correctly retrieves and displays target details
	Boundary (Edge Cases)	1. Click on a target with extremely long/short name E.g.: Ensure UI does not break, and backend handles the request 2. Click on a target when more than 100 targets exist in the system E.g.: Ensure performance remains stable and no lag occurs
	Erroneous (Invalid Input)	1. Click a deleted or non-existent target E.g.: API returns 404 Not Found and frontend shows error message 2. Backend fails to respond E.g.: System displays error message such as “Unable to load target details”
Expected result (test oracle)	The test **passes** if: • The correct target details are retrieved from the Targets table and correct information is displayed on the frontend. • The backend returns valid responses with correct data • System performance remains stable even with more than 100 targets • The UI remains functional with error handling in place The test **fails** if: • Incorrect, incomplete or no target details are retrieved from the Targets table and incorrect, incomplete or no information is displayed on the frontend. • API request is sent with the wrong targetID • The system does not handle non-existent target selection properly • UI crashes or fails to load target details • Unauthorized users can retrieve or manipulate target details they should not have access to	
Comments	Include test scenarios for users with no associated targets to verify proper handling of empty states.	
Created by	RDY	

Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18

Test 7

Test Case ID	**sys-test_02**	
Description of test	Verify that when a staff member has a target assigned to them, they receive automated monthly reminders.	
Related requirement document details	**Automated reminders (Priority: Must have)** *"As a manager, I want the system to send automated monthly reminders to project stakeholders so they can update their progress on the project and complete overdue tasks."*	
Prerequisites for test	- Access to the functional web-application - User account (contains valid email address) - At least one target must be assigned to the user - System's scheduler for reminders must be enabled	
Test procedure	1. Set the system's date to the last day of the month. 2. Trigger the automated reminder function via a POST API call 3. Verify that the system generates reminders for all staff members with assigned targets. 4. Check the system database to confirm that an email reminder was queued for each assigned user. 5. Validate that the email service function executes properly without errors.	
Test material used	- Functional web application - Database records with pre-registered staff and assigned targets - API request to trigger reminders - System logs to check processed reminders - Computer/mobile devices connected to the internet - Web browser (e.g. Google Chrome)	
Equivalence Classes	**Normal (Valid Input)**	Staff member has a valid email and assigned target E.g.: Reminder email is sent successfully and logged
	Boundary (Edge Cases)	1. Staff member has more than 100 targets assigned E.g.: Ensure only one email is sent and the system handles the load 2. Staff member does not have any assigned targets E.g.: No reminder should be sent 3. System must send 1000 reminders at once

		E.g.: System should not crash, and all reminders should be delivered
	Erroneous (Invalid Input)	1. Staff member does not have a registered email E.g.: System logs failure but does not crash 2. Staff member was deleted E.g.: Reminder should not be sent
Expected result (test oracle)	The test **passes** if: • All staff members with assigned targets receive an email reminder prompting them to update their progress • The system logs successful reminder dispatch for each valid user • No duplicate emails are sent for the same target • The reminder function does not trigger multiple times in one cycle • The UI and system logs display error messages properly if an issue occurs • The system does not crash The test **fails** if: • No email is sent to staff members despite valid targets being assigned. • Emails are sent incorrectly such as duplicate reminders and incorrect users. • The system crashes or takes too long to process reminders. • A staff member without an assigned target received an email	
Comments	Test across different browsers and devices Verify reminder email formatting across different email clients (Gmail, Outlook, etc)	
Created by	JJ	
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18	

Test 8

Test Case ID	**sys-test_03**
Description of test	Verify that when a valid userID is received to view the profile, the correct user details are retrieved from the User table and displayed on the Profile page.
Related requirement document details	**Login system (Priority: Must have)** *"As a user, I want to be able to create an account so that I can securely log in to the system that is tailored towards me."*
Prerequisites for test	- Access to the functional web-application - User logged into their account - User information such as name and profile picture must be available in the database

Test procedure	1. Navigate to the profile page by clicking on the profile icon in the top right corner. 2. Validate if the correct user information is retrieved and displayed on the Profile page 3. Compare the retrieved profile data with the Users table in the database	
Test material used	- Functional web-application - User accounts for staff member and manager - Database with User table containing stored profile data - Computer/mobile devices connected to the internet - Web browser (e.g. Google Chrome)	
Equivalence Classes	**Normal (Valid Input)**	User clicks on the profile icon E.g.: Correct profile details appear
	Boundary (Edge Cases)	1. User has no profile picture E.g.: Default avatar is shown 2. User has a very long name; more than 100 characters or special characters are used E.g.: UI handles display properly 3. User attempts to load another user's profile E.g.: Access is denied (403 Error)
	Erroneous (Invalid Input)	User manually enters an invalid userID in the URL E.g.: System returns "User not found" and an error message "Service unavailable" is displayed on the frontend
Expected result (test oracle)	The test **passes** if: • The correct user details are retrieved from the User table, and the user profile with correct user information is displayed on the Profile page. • Profile loads smoothly even with large datasets • System handles edge cases properly The test **fails** if: • Incorrect, incomplete or no user details are retrieved from the User table, and the user profile with incorrect, incomplete or no user information is displayed on the Profile page. • User is able to view another user's profile • System crashes or loads too slowly under high traffic	
Comments	Test across different browsers Test under various network conditions (slow WIFI, mobile data)	
Created by	EK	
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18	

<table>
<tr><td>Test Case ID</td><td colspan="2">sys-test_04</td></tr>
<tr><td>Description of test</td><td colspan="2">Verify that when two users update progress on the same target at the same time, the system enforces effective concurrency control.</td></tr>
<tr><td>Related requirement document details</td><td colspan="2">Ability to edit goals (Priority: Must have)
“As a staff member I want to have the ability to edit goals because I need to keep track of the targets I have been set.”</td></tr>
<tr><td>Prerequisites for test</td><td colspan="2">- Access to the functional web-application
- User accounts for at least two users
- A shared target that both users have access to</td></tr>
<tr><td>Test procedure</td><td colspan="2">1.User A logs in and navigates to a target page.
2.User B logs in simultaneously and navigates to the same target page.
3.User A updates progress and clicks "Save"
4.User B updates progress at nearly the same time and clicks "Save".
5. Verify how the system handles the concurrent updates
6. Check if any error messages are displayed and verify system logs for conflict handling
7. Verify that no data loss occurs, and the final saved progress is consistent</td></tr>
<tr><td>Test material used</td><td colspan="2">- Functional web application
- Database access logs to monitor conflicting changes
- Computer/mobile devices connected to the internet
- Web browser (e.g. Google Chrome)</td></tr>
<tr><td rowspan="3">Equivalence Classes</td><td>Normal (Valid Input)</td><td>Both users update different fields within the same target

E.g.: System correctly saves updates without conflicts</td></tr>
<tr><td>Boundary (Edge Cases)</td><td>User B submits changes milliseconds after User A

E.g.: System prevents data loss</td></tr>
<tr><td>Erroneous (Invalid Input)</td><td>One user enters invalid data such as an empty progress update or special characters

E.g.: System rejects invalid input and prevents overwriting valid data</td></tr>
<tr><td>Expected result (test oracle)</td><td colspan="2">The test passes if:

• The system correctly enforces concurrency control by locking the Targets table during User A’s edits, saving their changes, retaining a copy of the pre-edit state, applying User B’s changes to the original state, and resolving differences accordingly.
• Only valid data is saved

The test fails if:
• The system does not properly handle concurrency, leading to data loss, overwrites, or inconsistencies when both users attempt to save changes.
• System crashes
• Both users see different versions of the same target after saving</td></tr>
</table>

Comments	Test multiple users updating at the same time. E.g. 3-4 staff members editing simultaneously Verify if conflict handling is consistent across all devices
Created by	EK
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18

Integration Testing

Test 10

Test Case ID	**int-test_00**	
Description of test	Verify that when a user forgets their password, the password reset process only updates the password for valid users in the Users table.	
Related requirement document details	**Login System (Priority: Must have)** *"As a user, I want to be able to create an account so that I can securely log in to the system that is tailored towards me."*	
Prerequisites for test	- Access to the functional web-application - User account - Users table	
Test procedure	1. The user loads web-application 2. Click the forgotten password link, "forgotten password" page loads. 3. Enter 'Name', 'Email', and 'New Password' 4. Click the 'Submit' button, "Login" page loads 5. Enter username and new password 6. Access granted with new password	
Test material used	- Functional web-application - Database access logs for verification - Computer/mobile devices connected to the internet - Web browser (e.g. Google Chrome)	
Equivalence Classes	**Normal (Valid Input)**	Valid name and email given (matching in Users table) E.g.: Sets a new password and logs in successfully
	Boundary (Edge Cases)	1. Name and email given that is maximum character limit E.g.: User enters a 255-character email and 100-character name. System handles input correctly without errors 2. User enters special characters in the new password E.g.: User enters P@sSw0rd!#% as a new password. System accepts and updates the password

	Erroneous (Invalid Input)	1. Invalid name and email given (not matching in users database) E.g.: User enters email that is not registered, system displays an error message and does not reset the password 2. User enters multiple incorrect reset attempts E.g.: User tries five incorrect password reset requests, system locks account temporarily
Expected result (test oracle)	The test **passes** if: • With a valid name and email given, the password is updated in the Users table with the new password, allowing access with the new password. • With an invalid name and email given, the password is not updated in the Users table, the password remains the same. Error message shows with 'Wrong user information' The test **fails** if: • With a valid name and email given, the password is not updated in the Users table with the new password. The old password can still be used. • With an invalid name and email given, the password is updated in the Users table. • The system crashes.	
Comments	Ensure passwords are hashed for security purposes	
Created by	MC	
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18	

Test 11

Test Case ID	**int-test_01**
Description of test	Verify that when a user logs in successfully, all assigned targets are retrieved correctly from the Targets table and displayed on the frontend.
Related requirement document details	**Login system (Priority: Must have)** *"As a user, I want to be able to create an account so that I can securely log in to the system that is tailored towards me."* **Ability to edit or delete goals (Priority: Must have)** *"As a staff member I want to have the ability to edit goals because I need to keep track of the targets I have been set."*
Prerequisites for test	- Access to the functional web-application - User account - User must have targets assigned to them

Test procedure	1. User logs into their account 2. Validate if the correct target is retrieved from the database and displayed on the frontend.	
Test material used	- Functional web application - Database access logs - Computer/mobile devices connected to the internet - Web browser (e.g. Google Chrome)	
Equivalence Classes	**Normal (Valid Input)**	User has valid login credentials and assigned targets E.g.: User logs in successfully and all assigned targets are retrieved and displayed on the frontend
	Boundary (Edge Cases)	User has extremely high number of assigned targets E.g.: User logs in with more than 500 assigned targets. The system retrieves all targets correctly without lag or errors
	Erroneous (Invalid Input)	User logs in with invalid credentials and has no assigned targets E.g.: User has no assigned targets in the database. The system displays a “No targets assigned” message
Expected result (test oracle)	The test **passes** if: • All targets are correctly retrieved from the Targets table, and the frontend accurately displays all targets to the user. • If no targets are assigned, the system properly informs the user. The test **fails** if: • The targets are incorrectly or not retrieved from the Targets table, and the frontend either displays incomplete targets to the user, or no targets to the user. • The system crashes when handling a high number of assigned targets.	
Comments	Test with users having varying numbers of assigned targets to evaluate performance	
Created by	EK	
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18	

Test 12

Test Case ID	**int-test_02**
Description of test	Verify that when the targetID is received to edit the target, details are correctly updated in the Targets table.
Related requirement document details	**Ability to edit goals (Priority: Must have)**

	"As a staff member I want to have the ability to edit goals because I need to keep track of the targets I have been set."	
Prerequisites for test	- Access to the functional web application - Database access logs - User logged in to their account - targetID must be available in Targets table	
Test procedure	1. User retrieves existing target data by clicking on a target 2. Change details that need to be changed 3. Click 'Save' button, sending update request to backend 4. Verify that the API response confirms a successful update (HTTP 200 OK) 5. Retrieve the target from the database using the same targetID 6. Check details have updated correctly 7. Refresh the frontend and verify that the updated details are displayed accurately	
Test material used	- Functional web application - Database access logs - Computer/mobile devices connected to the internet - Web browser (e.g. Google Chrome)	
Equivalence Classes	**Normal (Valid Input)**	User edits an existing target and saves changes E.g.: User updates target details and clicks 'Save'. The backend responds with HTTP 200 OK, and changes reflect in the database and UI
	Boundary (Edge Cases)	User edits a target with maximum character limits E.g.: User modifies target name with 255 characters. The system successfully saves and displays the changes
	Erroneous (Invalid Input)	User tries updating a target that does not exist E.g.: User enters a non-existing targetID, the system returns an error message (HTTP 404 Not Found)
Expected result (test oracle)	The test **passes** if: • The database and frontend reflect the updated target details. • The backend provides a successful update response. The test **fails** if: • The frontend and backend do not reflect the updated target details. • The database is not rolled back (returned to a previous state). • The system does not handle concurrency issues properly.	
Comments	Test with multiple concurrent edits to verify conflict resolution	
Created by	TO	
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18 macOS Sequoia 15.0.1	

Test Case ID	**int-test_03**	
Description of test	Verify that filtering targets by a valid time frame displays the correct targets.	
Related requirement document details	**Filter tasks by period (Priority: Should have)** *"As a staff member, I would like to filter tasks by period (e.g. day, week, month, year) in order to know which tasks are due within a specific timeframe."*	
Prerequisites for test	- Access to the functional web-application - User account - Target table with targets having due dates in various time frames	
Test procedure	1. Log in to the system as an authenticated user 2. Under the "My Targets" section, select the desired time frame to filter tasks by. 3. Verify that only tasks due within the selected time frame are displayed	
Test material used	- Functional web-application - Database access logs - Computer/mobile devices connected to the internet - Web browser (e.g. Google Chrome)	
Equivalence Classes	**Normal (Valid Input)**	Valid time frame is given E.g.: User selects "This Week" filter, and only tasks due within the current week are displayed
	Boundary (Edge Cases)	Time frame selection at boundary limits E.g.: User selects "End of Month", and tasks due on the last day of the month are correctly included
	Erroneous (Invalid Input)	Invalid time frame given E.g.: User enters an incorrect format instead of a date range, and the system displays an error message without crashing
Expected result (test oracle)	The test **passes** if: • With a valid time frame given, only targets within that time frame are displayed. • With an invalid time frame given, no targets are displayed, and an error message is shown. The test **fails** if: • With a valid time frame given, incorrect targets are displayed. • With an invalid time frame given, targets are incorrectly displayed. System crashes.	
Comments	Test with different time zones to ensure consistency	
Created by	TO	
Test environments(s)	Windows 11 or Windows 10	

	Android 15 iOS 18

User Acceptance Testing

Test 14

Test Case ID	**ua-test_00**	
Description of test	Verify that the web application has a responsive layout across various devices and screen sizes, while maintaining full functionality and operating normally.	
Related requirement document details	**Responsive web design (Priority: Must have)** *"As a user, I want responsive web design on the project, so the system is accessible and responsive on various devices."*	
Prerequisites for test	- Access to the deployed web-application - User accounts - All devices should have the latest version of mainstream browsers installed.	
Test procedure	1. Open the web-application on a desktop computer to observe the rationality of the layout and the completeness of functionalities. 2. Adjust the browser window width to check for any misplacements or overflows of elements. 3. Access the web-application on a tablet device in both landscape and portrait modes to ensure interface elements are correctly rearranged. 4. Visit the web-application on a smartphone to verify if menus correctly fold and texts are readable. 5. Perform interaction tests, such as clicking buttons and filling out forms, to ensure all functionalities are accessible. 6. Record any layout disarray or functional anomalies.	
Test material used	- Test devices (desktop computers, tablets, and smartphones) and their specifications. - URL of the test web-application. - Browser compatibility testing tools. - Multiple different browsers (i.e. Google Chrome, Firefox, Safari)	
Equivalence Classes	**Normal (Valid Input)**	Standard screen sizes E.g.: Test on 1920x1080 (desktop), 768x1024(tablet), 357x667(mobile) to ensure elements adapt properly
	Boundary (Edge Cases)	Extremely small/ large screen sizes Test on a small phone with 320x480, 4K UHD (3840x2160) for layout distortion
	Erroneous (Invalid Input)	Unsupported browsers or outdated versions E.g.: Open in outdated Safari or text-based browsers and check for graceful degradation

Expected result (test oracle)	The test **passes** if: • The page will correctly adapt to different screen sizes without needing horizontal scrolling. • Interactive components (buttons, menus, forms, etc.) will function normally on all devices without any content misalignment or overflow. • The layout will be neat, with good readability and reasonable spacing. • The application degrades gracefully in unsupported browsers and does not crash The test **fails** if: • Potential issues may include improper adaptation of the layout during viewport changes • Interactive elements not functioning properly, or elements being truncated. • The system crashes or fails to display content correctly
Comments	N/A
Created by	HB
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18

Test 15

Test Case ID	**ua-test_01**
Description of test	Verify that the progress bar is functional and updates dynamically.
Related requirement document details	**Progress Bar (Priority: Must have)** *"As a staff member, I want a progress bar because I want to be able to view the progress of my goals visually."*
Prerequisites for test	- Access to the deployed web application - User logged in on their account - Targets with various progress percentages assigned. - At least one target where progress can be updated incrementally.
Test procedure	1. Locate a target with an existing progress percentage 2. Update the target's progress (e.g., mark sub-tasks related to the target as complete). 3. Observe the progress bar to confirm it updates accordingly. 4. Repeat steps for increasing progress levels (e.g., 25% → 50% → 75% → 100%). 5. Verify that the progress bar reaches 100% when the target has been fully completed.
Test material used	- Deployed web-application - A pre-loaded target dataset with varying progress levels. - Various devices - System logs (database updates, backend API responses and console logs)

Equivalence Classes	**Normal (Valid Input)**	Sub-tasks are marked as complete, progress updates dynamically E.g.: User marks progress of sub-task 1 as done. Progress bar increases between (25%, 50%, 75%, 100%)
	Boundary (Edge Cases)	Marking the last sub-task completes progress E.g.: User marks last sub-task progress. Progress bar moves from 0% to 100% in one step
	Erroneous (Invalid Input)	Trying to mark a completed subtask again E.g.: User marks an already completed sub-task, system prevents duplicate update
Expected result (test oracle)	The test **passes** if: • The percentage on the bar should update dynamically and the bar should reach 100% when all assigned sub-tasks have been completed. This should be reflected in the database as well. • Invalid actions are properly handled with error messages • System does not lag during multiple updates • Multiple users see the same progress changes in real-time The test **fails** if: • The progress bar does not update dynamically, and sub-task completions are not reflected in the database. • Concurrency issues occur • The progress bar lags in real-time updates	
Comments	Test concurrency by having multiple users update subtasks at the same time.	
Created by	TO	
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18 macOS Sequoia 15.0.1	

Test 16

Test Case ID	**ua-test_02**
Description of test	Verify that users find it easy to view their assigned targets.
Related requirement document details	**Clear and intuitive user interface (Priority: Must have)** *"As a user, I want the system design to be clear and intuitive so that it is easy to use and allows me to focus my full attention on my work."*
Prerequisites for test	- Access to the deployed web-application - User account - Targets table for a specific user. - The Targets table is updated with new target details. - Dashboard user interface is accessible for verification

<table>
<tr><td>Test procedure</td><td colspan="2">1. Log in to the system using the specific registered staff member’s credentials.
2. Navigate to the MyTargets section on the dashboard.
3. Verify the displayed target list is similar to the one in the database.
4. Modify or add an assigned target on the manager dashboard manually and reload the staff member’s dashboard.
5. Check if the staff member’s dashboard reflects the updated target in real time.</td></tr>
<tr><td>Test material used</td><td colspan="2">- Deployed web application
- Frontend dashboard user interface
- System database
- Manager dashboard for target modification
- Computer/mobile devices connected to the internet
- Web browser (e.g. Google Chrome)</td></tr>
<tr><td rowspan="3">Equivalence Classes</td><td>Normal (Valid Input)</td><td>A user has assigned targets and can view them

E.g.: A user logs in and sees all their assigned targets displayed correctly</td></tr>
<tr><td>Boundary (Edge Cases)</td><td>A user has an extremely high number of targets assigned

E.g.: More than 500 targets are assigned. The system should retrieve and display all targets without crashing or lagging</td></tr>
<tr><td>Erroneous (Invalid Input)</td><td>A user has no assigned targets

E.g.: The system should display a “No assigned targets” message</td></tr>
<tr><td>Expected result (test oracle)</td><td colspan="2">The test passes if:
The dashboard displays the correct list of targets with their respective details. When the database is updated, the lists of targets on the dashboard are also updated in real time.
If no targets are assigned, a “No assigned targets” message should be displayed.
System should remain responsive even when handling large numbers of targets.
The test fails if:
The dashboard does not show the expected list of targets.
The dashboard does not show the modified targets from the database after refreshing.
The user interface crashes or fails to load if database connectivity is lost.
An error message is shown on the dashboard to try again later.
System crashes or slows down when handling many targets</td></tr>
<tr><td>Comments</td><td colspan="2">Conduct usability testing to ensure ease of navigation and clarity of UI</td></tr>
<tr><td>Created by</td><td colspan="2">JJ</td></tr>
<tr><td>Test environments(s)</td><td colspan="2">Windows 11 or Windows 10
Android 15
iOS 18</td></tr>
</table>

Test Case ID	**ua-test_03**	
Description of test	Verify that the user finds it easy to use the filter function.	
Related requirement document details	**Filter tasks by period (Priority: Should have)** *"As a staff member, I would like to filter tasks by period (e.g. day, week, month, year) in order to know which tasks are due within a specific timeframe."*	
Prerequisites for test	- Access to the deployed web-application - User logged in on their account - Targets table has targets that have due dates in various timeframes. - Test environment with realistic target data. - Users (staff members) with varying levels of digital literacy.	
Test procedure	1. Demonstrate how users can filter targets by date (Day, Week, Month, Year, etc.). 2. Ask the staff member to apply filters to find targets due within specific periods. 3. Gather feedback on ease of use, expected behaviour, and usability of results. 4. Check if the staff member can successfully retrieve expected targets without external help.	
Test material used	- Deployed web application - Various office computers and devices connected to the internet and with access to the test environment. - A pre-loaded target dataset with various due dates (covering different user scenarios). - A step-by-step guide for users explaining how to apply the filters. - A feedback form for each staff member covering ease of use, accuracy of tasks displayed, system performance and any additional comments.	
Equivalence Classes	**Normal (Valid Input)**	User selects a valid time frame, and the correct list of tasks are displayed E.g.: User selects "This Week" and only views targets that are due this week
	Boundary (Edge Cases)	User selects an extreme time range or an overlapping period E.g.: User selects the "Last Month" filter, checking tasks due in both the last week of previous month and the first week of the current month.
	Erroneous (Invalid Input)	User enters an invalid input format E.g.: User does not input a date and uses a different format. System provides an error message.
Expected result (test oracle)	The test **passes** if: • Staff members confirm that filtering is intuitive and meets expectations • Staff members can find specific tasks with ease. • Error messages are properly displayed when an invalid input is used	

	• The system remains responsive even with large datasets and concurrent users The test **fails** if: • The staff members confirm faults within the filtering functionality, prompting adjustments to the user interface. • The staff members struggle to find filtered tasks • Users require external assistance to use the filtering feature • The system does not provide proper error handling messages when an invalid input is entered • The system crashes when there are multiple users trying to access the filtering function
Comments	Adjust user interface, if necessary, based on feedback (e.g., if users struggle with selecting date ranges).
Created by	TO
Test environments(s)	Windows 11 or Windows 10 Android 15 iOS 18 macOS Sequoia 15.0.1

Results

The following tables present the test results obtained so far. As our project is still in development and follows an Agile methodology, not all tests have been completed yet. Since features and components are still evolving, we will conduct regression testing throughout the development cycle to ensure that new changes do not introduce issues in previously tested functionality.

For any failed tests, regression testing will be carried out after fixes are implemented to confirm that the defects are resolved, and that overall system stability is maintained.

User Acceptance Testing will be conducted after the product has been fully developed. This phase will involve end-users testing the system, undertaking the task of applying user stories to ensure it meets their requirements and expectations. Feedback from UAT will be used to make any final adjustments before product handover.

Completed Tests		**Incomplete Tests**
Test ID	**Pass/Fail**	
unit-test_01	Pass	unit-test_00
unit-test_03	Pass	unit-test_02
sys-test_00	Fail	sys-test_01
sys-test_03	Pass	sys-test_02
int-test_00	Pass	sys-test_04
		int-test_01
		int-test_02
		int-test_03

TestID	unit-test_01
Expected result	The page changes from the Login page to the Sign-up page.
Pass/Fail	Pass
Actual Result	Whilst the user was on the Login page, the user clicked on the sign-up button and was directed to the Sign up page.
Comments	N/A

TestID	unit-test_03
Expected result	The user is relocated to the Login page, logging the user out of the account.
Pass/Fail	Pass
Actual Result	Whilst the user was on the Profile page, the user clicked on the Log Out button, and was directed to the Login page, logging the user out of the account
Comments	N/A

TestID	sys-test_01
Expected result	Each role will be correctly redirected to their respective dedicated pages after login, and the pages will display user information and functional options that match the expected permissions of the role. Incorrect login attempts should be properly recognised, and the system should provide appropriate error messages.
Pass/Fail	Staff Login - Pass Manager Login - Fail Overall Fail
Actual Result	User logged in with valid staff login details and the system correctly directed the user to their staff dashboard. User logged in with valid manager login details and the system incorrectly directed the user to a staff dashboard.
Severity	Very Serious and Infectious
Comments	The manager dashboard does not render when a manager logs in as it has not been correctly connected with the login page.

TestID	sys-test_03
Expected result	The correct user details are retrieved from the User table, and the user profile with correct user information is displayed on the Profile page.
Pass/Fail	Pass

Actual Result	Whilst the user was on the Dashboard page, the user clicked on their Profile icon, and the system directed them to the Profile page which showed the correct details.
Comments	N/A

TestID	**int-test_00**
Expected result	When a valid name and email is given, the password is updated in the Users table with the new password, allowing access with the new password. When an invalid name and email is given, the password is not updated in the Users table and the password remains the same. Error message shows with 'Wrong user information'.
Pass/Fail	Pass
Actual Result	Whilst the user was on the Forgotten Password page and entered a valid name and email with the new password, the password was correctly updated in the User table, and the user could log in with the new password. The user was not able to submit a new password after giving an invalid name and email.
Comments	N/A

Section 3

Testing Content

This section outlines the environment and tools required for testing. By specifying these, we ensure clarity on how the tests are executed, the conditions under which they are performed, and the dependencies that must be considered.

Device and Browser Diversity

Considering our target audience diverse range of users with varying device preferences, our testing environment includes different types of devices to ensure compatibility and usability.

Devices:

- **Desktop:** Windows 10 or 11, MacOS
- **Mobile:** Android, iOS (latest stable versions)

Browsers:

- **Google Chrome** (latest stable version, x64)
- **Safari** (latest stable version, x64)
- **Microsoft Edge** (latest stable version, x64)
- **FireFox** (latest stable version, x64)

The reason we choose these devices and browsers is that they dominate the global market share, ensuring our project can cover most of our users and meet client expectation. Testing across multiple environments helps identify browser-specific rendering issues, performance discrepancies, and user experience variations.

Testing Tools and Technologies

API Testing Tools: Postman ensures accurate validation of backend services and helps maintain API reliability, performance, and security of the system.

Load and stress testing: Use JMeter to help server crash by evaluating its performance under different levels of load.

Functional Testing: Cypress is a Functional Testing tool that can run and test automatically and more reliably since it eliminates human errors.

CI/CD Integration: GitHub Actions is an automation tool within GitHub that enables Continuous Integration and Continuous Deployment. When code is edited, all code changes are automatically tested before deployment to improve efficiency and reduce probability of error occurrence.

- Continuous Integration (CI):

- When a developer pushes code changes to GitHub
- GitHub Actions automatically triggers the CI pipeline
- The pipeline checks out the code
- Installs necessary dependencies
- Runs automated tests

- Continuous Deployment (CD):

- If all tests pass, the pipeline proceeds to deployment
- The application is built, and artifacts are created
- Code is automatically deployed to the production environment

- Error Handling:

- If any tests fail, the developer is notified
- The code doesn't proceed to deployment until issues are fixed
- This ensures only working code reaches production

Severity Level Classification

The failures within our testing system have been classified into the following Beizer severity levels:

Severity Level	**Explanation**	**Examples**	**Relevant Tests**
Mild	Minor UI/UX issues that do not affect **critical functionality**	Signup button not functioning, failure to retrieve user data	unit-test_01, unit-test_02, unit-test_03, sys-test_03,
Moderate	Issues affecting **usability and non-core features**	Inability to filter targets, inconsistent website layout across devices	int-test_03, ua-test_00, ua-test_01, ua-test_02, ua-test_03
Serious	Issues impacting **critical functionality**, although with **possible workarounds**	Target editing failures, incorrect details being displayed when target boxes are clicked	int-test_01, int-test_02

Very Serious	Issues impacting **critical functionality**, with **no workarounds available**	Automated reminder failure, targets not being displayed	sys-test_01, sys-test_02,
Extreme	Issues within the context of **system-wide failure**, **data loss**, **critical** security issues	Manager login failing, manager login redirecting to incorrect dashboard	unit-test_00, sys-test_00, sys-test_04
Infectious	Issues that trigger **cascading failures** (where **a failure in one system component** triggers **failures in successive system components**, potentially leading to **system-wide failures**)[1]	Failing “forgot password” function can **lock users out**, affecting authentication, incorrect dashboard redirection causes **unauthorised data access**, affecting data security	int-test_00,

The above severity levels are well-aligned to the functional requirements of our project, and show the priority in which faults within our system should be addressed and the ease of resolving the faults. **Infectious, extreme and very serious** faults should be given **immediate** priority, and **mild, moderate and serious** faults can be resolved much later, as they do not affect the core functionality of the system.

The UAT tests have been classified as **moderate** as the tests do not assess the core functionality of the system, but the usability and user’s experience. Though they have been defined at this severity level, our top priority is our client and their requirements. We will ensure that all issues are addressed in alignment with their expectations.

Bibliography

1. Chamberlin, B. W. (2022). How Failures Cascade in Software Systems. BYU ScholarsArchive. https://scholarsarchive.byu.edu/etd/9474/