Software Test Plan Document

Form Buster: Web-Based Registration, Approval, and Tracking

Team Members

Daniel Acosta (dacosta2022@my.fit.edu)
Christopher Demuro (cdemuro2022@my.fit.edu)
Alex Merino (amerino2022@my.fit.edu)
Luka Miodrag Starcevic (lstarcevic2022@my.fit.edu)

Faculty Advisor

Phillip Bernhard (pbernhar@fit.edu)

Version 1.0

Feb 24, 2025

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this test plan is to outline the specific test cases we will be using to validate the functionality of the Form Buster web app and underlying database systems. This document will provide a record of the test procedures we intend to use for ensuring that our software functions properly under all valid conditions and avoids any consequences when handling invalid conditions.

1.2 Scope

This test plan covers testing the user interface of the web app, as well as the database we are using to store forms, form templates and account data and the custom data structures that are used to store this data. All menus, functions, and buttons in the web app must be validated, and the database must accept and correctly process all valid input. All invalid input must be filtered before it reaches the database, either with processing to make it valid, or identification to discard it. Permissions of each class of user will be validated; Students, Faculty, and Administrators.

1.3 Objective

The objective of this document is to provide tests to expose any issues that a user might run into when attempting to use the application so that they can be fixed before the finalization of this product. These tests should also ideally allow problems that could lead to malicious exploitation of the software to be fixed so that it cannot be nefariously exploited.

2. Test Approach

2.1 Testing Levels

2.1.1 UI/UX Testing

The user interface of the web app will be tested to ensure that all functions that are wholly contained within the web app client function correctly and do not cause any visual or functional abnormalities.

2.1.2 Database Testing

The database will be tested to ensure that all data is properly processed and stored, and that no invalid data is entered into or given from the database during any query. Database testing will also ensure that all valid data is stored properly and returned during relevant queries.

2.1.3 Interaction Testing

The interactions between the web app and the database will be tested to ensure that the web app is making all proper queries to the database for necessary information, that the database is returning the proper data in response to the web app's queries, and that the web app is correctly utilizing the information it receives from the database. The email notification system will also be tested as part of this process due to its external nature in relation to the other system components.

2.2 Testing Methods

2.2.1 Manual/Incidental Testing

Manual testing will be conducted by the development team to solve minor issues in usability or functionality. This testing will be used while new features are being implemented in the codebase to ensure that new functions are functioning properly on a surface level and to fix minor bugs that crop up during the development process.

2.2.2 Automatic Testing

Automatic testing will be conducted by writing unit and module tests that test various scenarios within the web app and database to ensure they respond as intended. These will be repeatable tests that will generate reports and/or output to ensure that all functionality is working properly. These tests will be used to ensure that the software can handle all contingencies and unanticipated edge cases.

3. Test Items

The following items must be tested to complete the requirements outlined herein:

- User Authentication
- Profile Creation
- User Permissions Assignment/Management
- Form Creation/Completion
- Form Storage
- Form Tracking
- Form Denial/Approval
- Feedback
- Form Template Creation/Editing
- Form Template Management
- Web App Notifications
- Email Notifications
- Web App Configuration

4. Test Cases

4.1 User Authentication

4.1.1 User Login

Test Case 1: User enters correct username and password
Expected Outcome: User is logged in.

Test Case 2: User enters incorrect username.
Expected Outcome: An appropriate error message is displayed.

Test Case 3: User enters incorrect password.
Expected Outcome: An appropriate error message is displayed.

Test Case 4: User fails to fill out both fields:
Expected Outcome: An appropriate error message is displayed.

4.2 Dashboard Testing

4.2.1 In-Progress Form List

Test Case 1: Dashboard queries database for list of in-progress forms and receives valid data.
Expected Outcome: Dashboard receives this data and properly displays a list of the user's in-progress forms, both forms they initiated and forms they must approve/deny.

Test Case 2: Dashboard queries database for list of in-progress forms and receives invalid data.
Expected Outcome: Dashboard displays an appropriate error message and provides the user the option to re-query the database.

Test Case 3: Dashboard queries database for list of in-progress forms and receives no data.
Expected Outcome: Dashboard displays an appropriate error message and provides the user the option to re-query the database.

4.2.2 Form Tracking

Test Case 1: Dashboard queries database for tracking status of forms and receives valid data.
Expected Outcome: Dashboard receives this data and properly displays a list of the user's in-progress forms, both forms they initiated and forms they must approve/deny.

Test Case 2: Dashboard queries database for tracking status of forms and receives invalid data.
Expected Outcome: Dashboard displays an appropriate error message and provides the user the option to re-query the database.

Test Case 3: Dashboard queries database for tracking status of forms and receives no data.
Expected Outcome: Dashboard displays an appropriate error message and provides the user the option to re-query the database.

4.2.3 Notification Display

Test Case 1: Dashboard queries database for notification list and receives valid data.
Expected Outcome: Dashboard receives this data and properly displays the user's notification information.

Test Case 2: Dashboard queries database for notification list and receives invalid data.
Expected Outcome: Dashboard displays as if the user has no new notifications, and if further information is requested by the user, the Dashboard displays an appropriate error message.

Test Case 3: Dashboard queries database for notification list and receives no data.
Expected Outcome: Dashboard displays as if the user has no new notifications, and if further information is requested by the user, the Dashboard displays an appropriate error message.

4.2.4 Hyperlink/Button Testing

Test Case 1: User requests more detailed information about the status of a particular form.
Expected Outcome: Dashboard queries database for form information and loads appropriate status page in the web app.

Test Case 2: User navigates to their inbox.
Expected Outcome: Dashboard loads the inbox page in the web app.

Test Case 3: User navigates to the settings page.
Expected Outcome: Dashboard loads the settings page in the web app.

Test Case 4: User navigates to their profile page.
Expected Outcome: Dashboard loads their profile page in the web app.

Test Case 5: User navigates to the form creation page.
Expected Outcome: Dashboard loads the form creation page in the web app.

4.3 Form Creation Testing

4.3.1 Form Initialization

Test Case 1: User selects a form to create.
Expected Outcome: Web App loads the appropriate form template and displays it to the user.

Test Case 2: User cancels form creation.
Expected Outcome: Web App reloads the Dashboard.

4.3.2 Form Filling

Test Case 1: User fills in the form with valid data.
Expected Outcome: Form Creator accepts the data and prepares it for saving within the database.

Test Case 2: User fills in the form with invalid data.
Expected Outcome: Form Creator displays an appropriate error message informing the user that the data in relevant fields is invalid.

Test Case 3: User attempts to complete and submit form with required fields incomplete.
Expected Outcome: Form Creator displays an appropriate error message that the fields must be completed.

4.3.3 Form Submission

Test Case 1: User submits form with all data filled in correctly.
Expected Outcome: Form data is processed and saved in the database, and a confirmation message is displayed.

Test Case 2: User cancels form submission.
Expected Outcome: Form data is discarded and the user is taken back to the Dashboard.

4.4 Form Tracking

4.4.1 Tracking Status Updates

Test Case 1: User submits a form.
Expected Outcome: Form tracking status is initialized, providing a list of requirements for a form to be fully completed.

Test Case 2: User approves a form.
Expected Outcome: Form tracking status is updated to reflect the approval within the completion status.

Test Case 3: User denies a form.
Expected Outcome: Form is returned to origin with a comment explaining why the form was denied.

4.4.2 Form Tracking Notifications

Test Case 1: Form requires approval from a user.
Expected Outcome: User receives a notification in the web app Inbox and in their email that informs them of the actions required for the form.

Test Case 2: Form has been completed.
Expected Outcome: User receives a notification in the web app Inbox and in their email that informs them that the form has been completed.

Test Case 3: Form has been denied.
Expected Outcome: User receives a notification in the web app Inbox and in their email that informs them that the form has been denied and the stated reason for denial.

5. Test Environment

5.1 Minimum Hardware/Software Requirements

- **Desktop:**
  - **CPU:** Intel Core i3
  - **RAM:** 4GB
  - **Storage:** 500GB HDD
  - **Network:** Broadband (at least 20mbps)
  - **OS:** Windows 10, Ubuntu 20.04

- **Mobile:**
  - **OS:** Android 12, IOS 13

5.2 Recommended Hardware/Software Requirements:

- **Desktop:**
  - **CPU:** Intel Core i5
  - **RAM:** 8GB
  - **Storage:** 500GB SSD
  - **Network:** Broadband (at least 100mbps)
  - **OS:** Windows 11, Ubuntu 24.10
- **Mobile:**
  - **OS:** Android 15, IOS 18

5.3 Additional Software Requirements

- **Integrated Development Environment (IDE)**: Used for writing, testing, and debugging code
- **Database Management System:** MongoDB, used for storing and retrieving form data

6. Resource Requirements

6.1 Personnel:

1. **Desktop Frontend Developer:**
   - **Name:** Alex Merino
   - **Role:** Manages the development of the web app desktop UI, including rendering information retrieved from the database
   - **Responsibilities:**
     - Creating desktop GUI mockups
     - Testing for bugs in the desktop interface
     - Ensuring that the desktop UI is compatible, both functionally and aesthetically, with modern desktop browsers

2. **Mobile Frontend Developer:**
    - **Name:** Daniel Acosta
    - **Role:** Manages the development of the web app mobile UI, including rendering information retrieved from the database.
    - **Responsibilities:**
        - Creating mobile GUI mockups
        - Testing for bugs in the mobile interface
        - Ensuring that the UI is compatible, both functionally and aesthetically, with modern mobile browsers

3. **Database Manager:**
    - **Name:** Christopher DeMuro
    - **Role:** Administering the database that stores all form data and user information.
    - **Responsibilities:**
        - Creating and maintaining database
        - Debugging database errors
        - Ensuring all data is valid and filtered

4. **Data Structure Design:**
    - **Name:** Luka Miodrag Starcevic
    - **Role:** Creating custom data structures for storing form templates in the database
    - **Responsibilities:**
        - Designing form template data structures
        - Ensuring compatibility between web app and backend database
        - Designing filtering algorithms for invalid data

6.2 Documentation

- **User Manuals:** A guide for users on how to interact with the application.
- **Requirement Documents:** Detailed explanation of all project requirements and specifications.
- **Test Case Documents:** Outline for each and every test case of the software, including input values and expected results
- **Testing Reports:** Summaries of tests conducted and results recorded, as well as bugs detected and resolutions to those bugs

7. Risks and Contingencies

7.1 Potential Risks

1. **UI Usability Bugs**
    - **Risk:** Display bugs that cause the web app to display incorrect or misleading information
    - **Impact:** These issues complicate user interaction, and take precious development time to track and fix

2. **Database Performance**
    - **Risk:** The database may underperform during testing, including during queries and storage
    - **Impact:** This makes it more difficult to complete testing, and hampers the end-user experience due to slow response times
3. **Database Overflow/Corruption**
    - **Risk:** The database could become full or parts of the data stored within it may become corrupted due to hardware or software failure
    - **Impact:** This would make it impossible for users to reliably trust the software, and crucial information may be lost

7.2 Mitigation Strategies

1. **UI Usability Bugs**
    - The UI will be regularly tested to ensure that all features are functioning properly
    - The UI will be of a modular design that will make it easier to track down and repair any issues that come up during development
2. **Database Performance**
    - The database will be load tested to ensure it performs adequately at all levels of load
    - The database query structure will be optimized to maximize performance
3. **Database Overflow/Corruption**
    - Regular and automatic backups of the database will be conducted
    - Monitoring alerts will be sent to Administrators to warn them of potential disk overflow

8. Success Criteria

The system will be considered ready for use when the following conditions have been met:

8.1 All Functional Requirements Verified By Test Cases

- The functional requirements outlined in the Requirement Document must be tested and verified using the above test cases.
    - **User Interface Testing:** The user interface must function correctly on all client devices and correctly perform all of its intended functions
    - **Database Testing:** The database must reliably store all form data and return it properly when relevant queries are made
    - **Notification Testing:** Notifications must be disseminated to the relevant users whenever the appropriate conditions are met
- All features must pass all relevant test cases without any failures or unexpected behavior

8.2 No Critical Bugs or Errors Remain

- All bugs during the testing phase must be catalogued and categorized by severity
- No critical bugs may be allowed to remain in the finalized system
- Critical bugs constitute the following:
    - Significant UI problems that prevent proper use of the application
    - Improper storage and retrieval of the database
    - Notifications not functioning or functioning intermittently

8.3 User Feedback is Overall Positive

- Test users must be satisfied with the experience of using the application
- Test users must be able to navigate the interface with little instruction, relying largely on their intuition
- Test users should not experience significant loading times or performance-related delays in loading application pages, using the application, retrieving data from the database, or receiving notifications from the application

9. Test Deliverables

The following deliverables will be provided upon completion of the testing process:

- **Test Cases:** The content and procedures of the test cases used in the validation process will be provided and thoroughly documented
- **Test Results:** Test logs, reports, screenshots, and any other associated documentation used in reproducing, debugging, or otherwise attempting to recreate or resolve a system issue will be compiled
- **Bug Reports:** Any issues that may be discovered will be catalogued and recorded along with detailed documentation on how to reproduce the issue and how it might be resolved