**A PROPOSED OFFERING OF A CLINIC RECORDS MANAGEMENT SYSTEM**

**FOR**

**HI-PRECISION DIAGNOSTICS – MALABON BRANCH**

A Thesis Project Presented to the

Faculty of Datamex College of Saint Adeline, Inc.

In Partial Fulfillment of the Requirements for the

Degree of Bachelor of Science in Information Technology

By:

Catubay, Mark Lawrence L.

**TESTING DOCUMENT**

**CHAPTER I**

**INTRODUCTION**

This chapter details the testing plan and results for the Clinic Records Management System (CRMS). The primary purpose of this testing phase is to systematically verify and validate that the application meets the functional, performance, and security requirements outlined in the design documentation. This process is critical for identifying defects, ensuring software quality, and confirming that the system is ready for deployment.

**Objectives:**

* To validate that all CRUD (Create, Read, Update, Delete) operations for the Patient, Consultation, and Medicine modules function as expected.
* To verify that the user authentication system correctly grants access to authorized users and denies access to unauthorized users.
* To ensure that the data search functionalities are accurate and responsive.
* To identify and document any bugs, defects, or deviations from the expected behavior.
* To confirm that the user interface is intuitive and the system is stable during normal operation.

**Scope of Testing**

The scope of this testing phase is focused on the core application logic and user interface as it runs in a development environment (web browser and local server).

**In Scope**

* User Authentication Module (Login and Logout).
* Patient Information Module (Full CRUD and Search).
* Consultation History Module (Adding and Viewing Consultations).
* Medicine Inventory Module (Full CRUD and Search).
* Basic UI responsiveness and data presentation.

**Out of Scope**

* Formal performance and stress testing under heavy load.
* Advanced security vulnerability and penetration testing.
* Testing of the final packaged Electron application installer and its specific desktop integrations (e.g., installation process, offline behavior). This will be conducted in a separate, post-packaging testing phase.

**CHAPTER II**

**TESTING ENVIRONMENT**

**Hardware Specifications**

* **Processor:** Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz, 2496 Mhz, 2 Core(s), 4 Logical Processor(s)
* **RAM:** 8 GB
* **Operating System:** Microsoft Windows 10 IoT Enterprise LTSC
* **Storage:**  237.86 GB (255,401,988,096 bytes)

**Software Requirements**

* **Runtime Environment:** Node.js v22.16.0
* **Package Manager:** npm v10.9.2
* **Web Browser:** Brave Browser
* **Code Editor:** Visual Studio Code v1.103.2
* **Database:** SQLite 3

**Test Data**

Sample data was created to simulate a realistic clinic environment. This includes:

* **User Accounts:** An administrator account with username admin and password 123.
* **Patient Records:** A set of 15 mock patient profiles with varied names, dates of birth, and addresses
* **Medicine Records**: A list of 20 common medicines with varying quantities.

**CHAPTER III**

**TESTING METHODOLOGY**

The primary testing approach used was **Black-Box Testing**. Each feature was tested from an end-user perspective, focusing on inputs and outputs without knowledge of the internal code structure. The process involved executing predefined test cases and comparing the actual results against the expected outcomes.

The main tool used for testing was the **Developer Tools**, specifically the **Network** tab to monitor API requests/responses and the **Console** tab to check for any client-side JavaScript errors.

Testing criteria were straightforward: a test case is marked as **"Pass"** if the actual output perfectly matches the expected output without any errors. It is marked as **"Fail"** if there is any deviation, which is then documented in the Bug Tracking log.

**CHAPTER IV**

**TEST CASES**

**This section provides a detailed breakdown of all test cases executed to verify the functionality of the Clinic Records Management System. Each case includes the module being tested, pre-conditions, specific steps, and the expected outcome.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Module | Test Description | Pre-conditions | Test Steps | Expected Output | Actual Output | Status | Remarks |
| Authentication & Session |  |  |  |  |  |  |  |  |
| TC-001 | Authentication | Successful login with valid credentials. | User is on the login page. A valid user ('admin') exists in the database. | 1. Enter username: 'admin'.2. Enter the correct password.3. Click the 'Login' button. | User is authenticated. The session flag is set. The user is redirected to dashboard.html. | User successfully redirected to the dashboard. | Pass | Core functionality is working. |
| TC-002 | Authentication | Attempt login with a valid username but an invalid password. | User is on the login page. | 1. Enter username: 'admin'.2. Enter an incorrect password.3. Click 'Login'. | The system displays an error modal/alert: "Wrong credentials!". The user remains on the login page. | Error modal appeared as expected. | Pass | Backend correctly rejects invalid passwords. |
| TC-003 | Authentication | Attempt login with a non-existent username. | User is on the login page. | 1. Enter username: 'invalid\_user'.2. Enter any password.3. Click 'Login'. | The system displays an error modal/alert: "Wrong credentials!". The user remains on the login page. | Error modal appeared as expected. | Pass | System correctly handles non-existent users. |
| TC-004 | Authentication | Attempt login by bypassing frontend validation for required fields. | User is on the login page. | 1. Use browser developer tools to enable the 'Login' button.2. Click 'Login' with both fields empty. | The API should return a 400 Bad Request with the message: "Username and password are required." | API returned 400 with the expected message. | Pass | Backend validation is robust. |
| TC-005 | Session | Successful user logout. | User is logged into the system. | 1. Click the 'Logout' link in the sidebar.2. In the confirmation modal, click 'Yes'. | The sessionStorage is cleared. The user is redirected to the login page (index.html). | User was successfully logged out and redirected. | Pass | N/A |
| TC-006 | Session | Attempt direct URL access to a protected page without logging in. | User is not logged in. sessionStorage is empty. | 1. Open a new browser tab.2. Navigate directly to patients.html. | The page script detects no active session and immediately redirects the user to index.html. | User was correctly redirected to the login page. | Pass | Session gatekeeper works as intended. |
| Patient Management |  |  |  |  |  |  |  |  |
| TC-007 | Patient | Add a new patient with complete, valid data. | User is logged in and on the 'Patients' page. | 1. Click 'Add New Patient'.2. Fill in all fields (Name, DOB, Address).3. Click 'Submit'. | The modal closes. The patient list refreshes, showing the newly added patient. | New patient appeared in the table. | Pass | N/A |
| TC-008 | Patient | Attempt to add a patient with missing required data (Name). | User is logged in and on the 'Patients' page. | 1. Click 'Add New Patient'.2. Leave the 'Name' field empty.3. Fill in the 'DOB' field.4. Click 'Submit'. | The API returns a 400 error. An alert appears with the message: "Error! Name and Date of Birth are both required." | API returned 400 with the correct error message. | Pass | Backend validation for patients is working. |
| TC-009 | Patient | Update an existing patient's information. | User is logged in and on the 'Patients' page. An existing patient record is visible. | 1. Click the 'Edit' button for a patient.2. In the modal, change the patient's address.3. Click 'Submit'. | The modal closes. The patient list refreshes, reflecting the updated information. | Address was updated successfully. | Pass | N/A |
| TC-010 | Patient | Delete an existing patient record. | User is logged in and on the 'Patients' page. | 1. Click the 'Delete' button for a patient.2. In the confirmation modal, click 'Yes'. | The patient record is removed from the list. The UI refreshes. The patient's consultation history is also deleted from the database (cascade delete). | Patient was removed from the UI. Verified in the DB that consultations were also removed. | Pass | Cascade delete is functional. |
| TC-011 | Patient | Search for an existing patient by name. | User is logged in and on the 'Patients' page. Multiple patients exist. | 1. Type the first few letters of an existing patient's name into the search bar. | The patient list dynamically filters to show only the records that match the search query. | The list filtered correctly and instantly. | Pass | N/A |
| Consultation Management |  |  |  |  |  |  |  |  |
| TC-012 | Consultation | Add a new consultation record to a patient. | User is logged in. A patient record exists. | 1. On the 'Patients' page, click 'View' for a patient.2. Fill in the 'Add New Consultation' form (Date, Complaint).3. Click 'Add Consultation'. | The new consultation appears immediately in the "Patient History" list within the modal. The form fields are cleared. | Consultation was added and displayed instantly. | Pass | N/A |
| TC-013 | Consultation | Attempt to add a consultation with missing required data (Complaint). | User is logged in and viewing a patient's consultation history. | 1. In the consultation modal, enter a date but leave the 'Complaint' field empty.2. Click 'Add Consultation'. | The API returns a 400 error. An alert appears with the message: "Both complaint and consultation date data are required." | API returned 400 with the expected error message. | Pass | Backend validation for consultations is working. |
| Medicine Inventory |  |  |  |  |  |  |  |  |
| TC-014 | Medicine | Add a new medicine with a unique name. | User is logged in and on the 'Inventory' page. | 1. Click 'Add New Item'.2. Enter a unique medicine name and a quantity.3. Click 'Submit'. | The modal closes. The new medicine appears in the inventory list. | New medicine was added successfully. | Pass | N/A |
| TC-015 | Medicine | Attempt to add a medicine with a name that already exists. | A medicine named 'Paracetamol' already exists in the inventory. | 1. Click 'Add New Item'.2. Enter 'Paracetamol' as the name.3. Enter a quantity.4. Click 'Submit'. | The API returns a 500 Internal Server Error with a message indicating a UNIQUE constraint violation (e.g., SQLITE\_CONSTRAINT: UNIQUE constraint failed: medicines.name). | API returned a 500 error with the expected constraint failure message. | Pass | Database schema constraint is enforced. |
| TC-016 | Medicine | Update the quantity and description of an existing medicine. | User is logged in and on the 'Inventory' page. | 1. Click 'Edit' for a medicine.2. Enter a new quantity and/or description.3. Click 'Submit'. | The modal closes. The inventory list refreshes, showing the updated information for that medicine. | Medicine details were updated correctly. | Pass | N/A |

**CHAPTER V**

**BUG TRACKING & ISSUE LOG**

During the testing phase, the following issues were identified.

**Table 9.2: Bug Tracking Log**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bug ID | Description | Severity | Reported By | Status | Resolution |
| B001 | The search bar does not automatically clear its text when the user navigates away from the Patients page and then returns. The old search term persists. | Low | Proponent | Open | Pending fix |
| B002 | The logout button incorrectly navigates to index.html instead of triggering the confirmation prompt first on certain pages. | Medium | Proponent | Resolved | The event listener was updated in script.js to correctly handle the click event and show a confirmation before redirecting. |

**CHAPTER VI**

**USER ACCEPTANCE TESTING (UAT)**

**Planned Test Scenarios for End-Users**

End-users will be given the following scenarios to perform without direct guidance, allowing for an authentic assessment of the system's usability.

* **Scenario 1- New Patient Registration and First Visit**
  + **Task:** A new patient has just arrived. Register them in the system. They are complaining of a headache. Record this visit.
  + **Steps:**
    1. Log into the system.
    2. Navigate to the 'Patients' page.
    3. Add the new patient with their provided details.
    4. Find the patient you just created.
    5. Open their record and add a new consultation entry for today's date, noting their complaint.
* **Scenario 2 - Existing Patient Follow-up and Inventory Update**
  + **Task:** An existing patient has returned for a follow-up. Find their record, review their last visit, and add a new consultation. They were given one box of a specific medicine. Update the inventory.
  + **Steps:**
    1. Navigate to the 'Patients' page and use the search bar to find the patient.
    2. View their consultation history to review their previous diagnosis.
    3. Add a new consultation entry for their follow-up visit.
    4. Navigate to the 'Inventory' page.
    5. Find the medicine that was dispensed and update its quantity.
* **Scenario 3 - Inventory Management - Stock Arrival**
  + **Task:** The clinic has just received a new shipment of medical supplies. Add a brand-new medicine to the inventory and update the stock for an existing one.
  + **Steps:**
    1. Navigate to the 'Inventory' page.
    2. Add a completely new medicine that is not on the list.
    3. Find an existing medicine on the list and update its quantity to reflect the new stock.

**UAT Feedback Form**

After completing the scenarios, each user will be asked to fill out a simple feedback form to capture their experience.

|  |  |
| --- | --- |
| UAT Feedback Form |  |
| Tester Name: |  |
| Date: |  |
| Scenario Tested: |  |
| Were you able to complete the task successfully? | Yes / No |
| On a scale of 1 (Very Difficult) to 5 (Very Easy), how easy was the system to use for this task? | 1 - 2 - 3 - 4 - 5 |
| Did you encounter any errors or confusing parts? If so, please describe them. |  |
| Do you have any suggestions for improvement? |  |

**CHAPTER VII**

**CONCLUSION & RECOMMENDATIONS**

**Summary of Test Results**

The testing phase for the web-based version of the CRMS was successful. All core functionalities related to patient management, consultations, medicine inventory, and user authentication are operating as designed. The system is stable, and the backend API correctly handles data validation and processing. A total of 11 test cases were executed, with all core features passing their functional requirements.

**Key Observations and Insights**

* The system's three-tier architecture has proven to be robust, with clear separation between the UI, business logic, and data layers.
* The real-time search functionality is fast and responsive with the current test data set.
* The backend API's error handling is consistent, providing clear error messages to the frontend.

**Recommendations:**

1. **Resolve Open Bugs -** It is recommended to resolve the identified bugs (B001) prior to the final packaging of the application.
2. **Proceed with Deployment Packaging -** The application is stable and ready to proceed to the next stage: packaging into a standalone desktop application using Electron.js.
3. **Conduct Post-Packaging Testing -** Once the application is packaged into an executable installer, a new, targeted round of testing must be conducted. This will focus on the installation process, offline functionality, and ensuring the database path and server initialization work correctly in a deployed environment.

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Changes |
| 1.0 | September 4, 2025 | Catubay, Mark Lawrence L. | Initial draft of the Testing Document. Pre-deployment |