**Test Plan for Nursing Patient Filtering Application**

**1. Introduction**

This test plan outlines the testing strategy for the **Nursing Patient Filter Application**, to help ensure that the functions work as expected and functions correctly, and also meets the requirements, of our client Utah County Health department.

**2. Test Objectives**

* Validate that the system is collecting data from according to the categories provided.
* Ensure data can be assigned to Nurses by zip code correctly.
* Test the user interface for usability and performance.
* Ensure old data can be accessed for editing and adjustments.
* Flag for duplicate patients when found along with a date of the first occurrence
* Verify compliance with healthcare data regulations ( HIPAA).
* Ensure system stability under expected and peak loads.

**3. Scope**

**3.1 Features to be Tested**

Patient filtering based on:

* Name
* Last Name
* Mothers Name
* Mothers Last Name
* DOB
* Address
* Zip Code
* Age

User roles and access control:

* Nurse director, Nurse Manager, Nurse Supervisor, Office Specialist

Data import and export:

* CSV/Excel file handling
* Secure API data retrieval
* Old data should be accessible to help identify duplicate patients

Error handling and security:

* Invalid data input handling
* Data encryption and protection

**3.2 Features NOT to be Tested**

Will not have an external database integration (assumed functional)

**4. Test Strategy**

**4.1 Testing Levels**

1. **Unit Testing**
   * Verify individual functions, such as filtering logic(By, Zipcode, city, etc)
2. **Integration Testing**
   * Ensure correct data flow between modules (e.g., from lists to → filter engine → UI).
3. **System Testing**
   * Validate the complete system's filtering accuracy and user workflows.
4. **User Acceptance Testing (UAT)**
   * Present to Nursing director to help validate the filtering results and usability.

**5. Test Cases**

**5.1 Functional Test Cases**

| **ID** | **Test Case** | **Input** | **Expected Output** | **Pass/Fail** |
| --- | --- | --- | --- | --- |
| TC001 | Filter by age | Age < =4 | All children below the age of 4 up to weeks or month old |  |
| TC002 | Filter by zipCode  And address | Utah county(only) | Children and their families for that region. |  |
| TC003 | Flag duplicates | Duplicates | Duplicate patients and families should have the first date it was found and some way to flag duplicate. |  |
| TC004 | Print Duplicates | Combined list or patients | Duplicates should have a way to export or print the list. |  |
| TC005 | Reload all lists for editing | Old patients lists | Retrieve old lists into the app for filtering purposes(finding old patients assigned Nurses etc). |  |

**5.2 Security Test Cases**

| **ID** | **Test Case** | **Expected Outcome** | **Pass/Fail** |
| --- | --- | --- | --- |
| ST001 | Unauthorized access attempt | Access denied |  |
| ST002 | Encryption | Lists should not be readable. |  |
|  |  |  |  |

**5.3 Performance Test Cases**

| **ID** | **Test Case** | **Test Method** | **Expected Outcome** | **Pass/Fail** |
| --- | --- | --- | --- | --- |
| PT001 | Load test (1000 patients) | Simulate large dataset | System filters within 3 seconds |  |

**6. Tools and Resources**

* **Testing Framework:** Selenium, PyTest (for automation)
* **Security Scanner:** OWASP ZAP , Postman Security tests, SoapUi, REST- Assured for vulnerability scanning
* **Performance Tools:** JMeter for load testing, Locust, Gatling

**7. Risk Analysis**

| **Risk** | **Mitigation** |
| --- | --- |
| Incorrect filtering logic | Perform unit testing |
| Data privacy breaches | Implement encryption and role-based access |
| If System crashes under load | Conduct performance and stress testing |

**8. Test Schedule**

| **Phase** | **Start Date** | **End Date** |
| --- | --- | --- |
| Test Planning | 2 days | 10 days |
| Test Case Development | 2 days | 5 days |
| Test Execution | 7 days | 12 days |
| Bug Fixing & Retest | 10 days | 15 days |
| UAT | 12 days | 17 days |

**4.1 Testing Levels**

**Unit Testing (White-Box)**

* **Approach:** White-box testing
* **What should be tested?**
  + Verify individual functions, such as filtering logic and input validation.
  + Check how the filtering algorithm processes inputs
  + Ensure correct handling of edge cases (empty records, invalid inputs).
  + Use **code coverage tools** to check execution paths.

**Integration Testing (Gray-Box)**

* **Approach:** Combination of black-box and white-box testing
* **What should be tested?**
  + Ensure correct data flow between modules (patient List**→ filter app → UI**).
  + Validate security aspects like role-based access control (**security model** that **restricts system access** based on a user's **role** within an organization).

**System Testing (Black-Box)**

* **Approach:** Black-box testing
* **What should be tested?**
  + Validate the **complete system's** filtering accuracy and efficiency
  + Test patient search and filtering functionality without looking at the internal implementation.
  + Ensure that different roles (nurse, admin) receive only the appropriate patient data.

**User Acceptance Testing (UAT) (Black-Box)**

* **Approach:** Black-box testing
* **What should be tested?**
  + Have **nurses and administrators** validate the filtering results for usability and correctness.
  + Conduct real-world scenario testing to ensure the system is robust and meets client requirements.
  + Get feedback on ease of use and UI design.