Software Test Plan Of RxCheck Application

Version 1.0.0

Prepared by:
Peter Butler
Richard Espinal
Colton Lancsarics
Guilherme Machado
Roselle Paala

Table of Contents

1.1 Goals and Objects 1

1.2 Intended Audience 1

1.3 Statement of Scope 1

1.4 Major Constraints 2

1.5 References 2

2.0 Testing Plan 2

2.1 Architecture Diagram 2

2.2.1 Content Testing 3

2.2.2 Database Testing 3

2.2.3 User Interface Testing 3

2.2.10 Implementation Testing 5

2.2.11 Testing Resources and Staffing 5

3.0 Testing Procedures 6

3.1 Test Tracking 6

3.2 Content Navigation 7

3.3 Database Testing 9

3.4 User Interface and Component Level Testing 9

3.5 Configuration Testing 10

3.6 Performance Testing 11

3.7 Implementation Testing 11

3.8 Security Testing 11

3.9 Standards Testing 12

1.0 Introduction

This document gives an overview of the Test Specification for the RxCheck web application. This document will contain details on how testing will be carried out and by whom testing will be conducted by.

1.1 Goals and Objects

RxCheck has serious real world use and implementations. RxCheck should be reliable all the time as it should be the new primary paperless system for doctors and healthcare providers. This document will be used with Agile methodologies and scrum teams. Teams will be used and divided up throughout testing to hone in on specific areas and have clear separation between testing components.

This document highlights the main paths, flows, boundaries, and limits of the software. This document will only account for expected use, where the testing teams will be encouraged to go outside of this document to test and try erroneous operations and try to error out the system.

During testing phases, the RxCheck application will monitor how users are interacting with the system by acquiring written and verbal feedback as well as logging within the application itself. This will help in future development and the performance of the application as it is maintained in the future.

1.2 Intended Audience

This document is intended for the programmers and testers of the RxCheck application. This document will describe each test that will be ran and how they will be carried out, hitting all corners of the application.

1.3 Statement of Scope

To achieve the goal of the final product there are many different tests RxCheck must go through. Our testing plan will be based off on the Webapp testing strategy. There are different cycles to this plan that will be divided up between teams, so that each team can focus on their specific area. If there are not enough teams or people, teams will switch between one or more of the testing cycles each sprint. The different levels for web app testing can be found below.

Content Testing
Database Testing
User Interface Testing
Component Level Testing
Navigation Testing
Configuration Testing
Performance Testing
Security Testing
Standards Testing

Implementation Testing

1.4 Major Constraints

The product will be tested on a smaller scale than that which it will be implemented in. This product will be rigorously tested pre-release for stress and performance issues but will not be tested at a higher level until a facility plans to use and implement RxCheck. RxCheck's first installation will allow the team to test at a higher more production like level before it will go live.

1.5 References

To thoroughly understand the concept of this documentation, please refer to the following documents:

- Software Requirement Specification of RxCheck Application. This will clarify the purpose of the RxCheck Application and how this app will be carried out.
- User Interface Diagram, User Interface Wireframes, and UML diagrams discussed in Software Design Documentation of RxCheck Application. This document will explain the architecture, component design, and user interface of RxCheck application.

2.0 Testing Plan

2.1 Architecture Diagram

Fig 2.1 illustrates the high-level architecture diagram of RxCheck Application. Please refer to Software Design Documentation for full details.

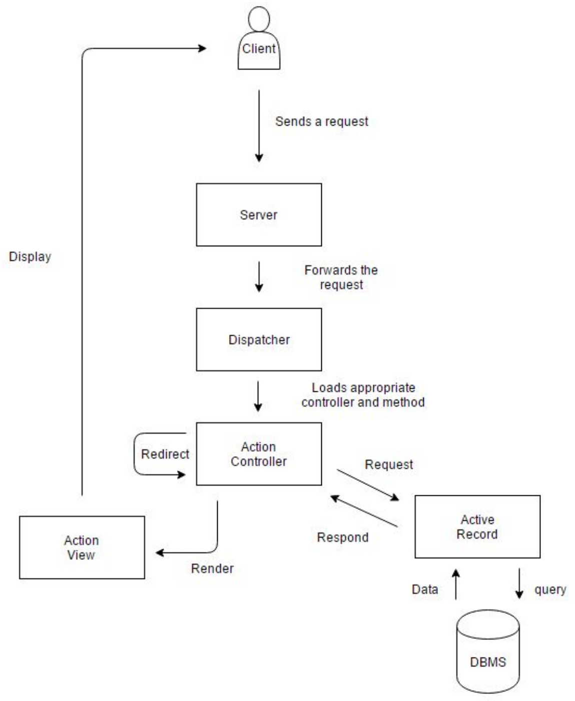


Fig 2.1 Model-View-Controller Architecture(MVC)

2.2 Testing Strategy

This section will provide the strategic steps that will be used in RxCheck Application.

2.2.1 Content Testing

Content testing evaluates the overall quality of the RxCheck web pages. Due to the critical nature of RxCheck Application's function, aside from the fact that the application must be appealing and customer-friendly, it is imperative that web pages populated by RxCheck Application are visually friendly, syntactically accurate, and the data presented are reliable and consistent. The following factors are some of things to consider when conducting content testing:

- Check for browser compatibility for font style
- Check for consistency of font style
- Check if the graphics used in the web page(s) is appropriate

2.2.2 Database Testing

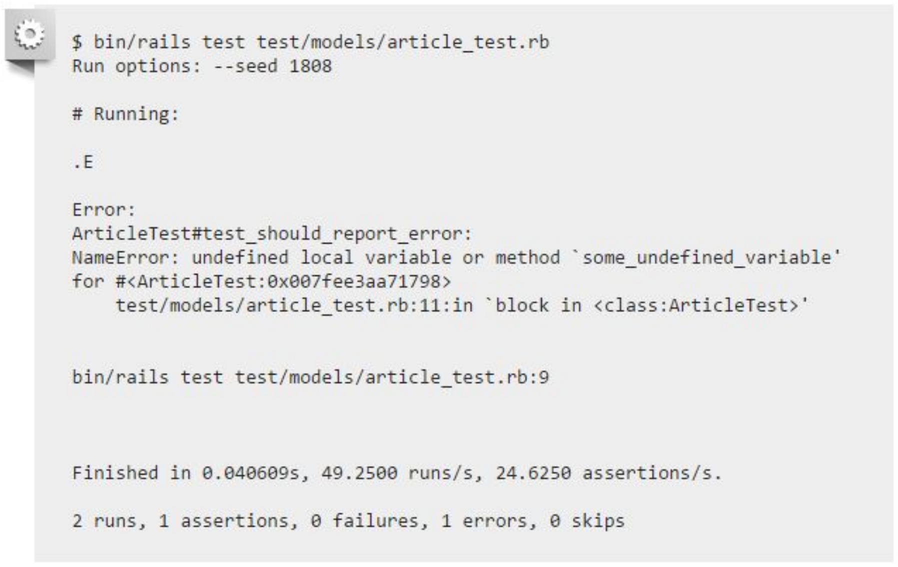
Database testing consists of testing the connection strings and connection to and from various databases. Along with this, the application must also be able to send, create, manipulate, delete, and or receive data from the database.

2.2.3 User Interface Testing

User Interface Testing and Content Testing go together. The aim of User Interface Testing is to put RxCheck web pages into real use. Healthcare providers, doctors, pharmacists, and administrators will be invited to test a website connected to RxCheck production database. This part of the test will validate if the user can login from the main page and move to the desired page with ease. For instance, is the user able to enter the username and password in the login page with ease? Does the webpage offer verification and additional validation on data entry before saving or deleting? Are there clear instructions for the user to navigate the web page? For a complete diagram of user interface wireframes and error/exception processing, please review the Software Design Documentation as mentioned in references.

2.2.4 Component Level Testing

Component level testing will begin at the developers when they go to run and debug their code. That is the first step in assuring that everything works syntactically and runs. Ruby on Rails can make a test environment where test cases for each module will be created and ran. Once created this will save developers a lot of time and give them valuable information such as runs, errors, failed runs, and time taken just to name a few. Below is an example of what an output would consist of from these tests.



```
$ bin/rails test test/models/article_test.rb
Run options: --seed 1808

# Running:

.E

Error:
ArticleTest#test_should_report_error:
NameError: undefined local variable or method `some_undefined_variable'
for #<ArticleTest:0x007fee3aa71798>
    test/models/article_test.rb:11:in `block in <class:ArticleTest>'


bin/rails test test/models/article_test.rb:9



Finished in 0.040609s, 49.2500 runs/s, 24.6250 assertions/s.

2 runs, 1 assertions, 0 failures, 1 errors, 0 skips
```

2.2.5 Navigation Testing

The navigation testing plan will consist of following the User Interface Sequence Diagram. Test Cases will be created for each path that the sequence could possibly follow. If multiple paths can be taken a test plan will be created for each. The tester will ensure that each path completes successfully and the desired node is reachable. These test cases can be integrated with the User Interface Testing and Content Testing, and will be tested during normal functionality testing as well.

2.2.6 Configuration Testing

The configuration testing plan will involve testing the RxCheck Web Application across most of major Web browsers, Google Chrome, Internet Explorer, Mozilla Firefox, Safari, and Opera. Following this approach the tester (s) will ensure that the RxCheck application can be accessed and operated on the different browsers that users may use to access RxCheck. These browsers were selected per their market share and popularity.
Per www.netmarketshare.com as of May 2016 the market share for each of the browsers selected

as test platforms are as follows.

MONTH	**CHROME**	**INTERNET EXPLORER**	**FIREFOX**	**MICROSOFT EDGE**	**SAFARI**	**OTHER**
May, 2016	**45.63%**	**33.71%**	**8.91%**	**4.99%**	**4.69%**	**2.07%**

In selecting these browsers as test platforms will ensure that during configuration testing, the errors encountered will reflect those errors most likely to be encountered in real use of the application and corrected before the RxCheck application goes live. In addition, testing will also be done for the mobile device versions of these browsers. Planning the Configuration Testing this way, will keep the number of possible configurations combinations low, which will speed the Configuration testing phase.

2.2.7 Performance Testing

The performance testing plan will be based around the response time of the RxCheck under different load conditions. The tester (s) will use pre-developed scripts to make a large amount of web request to RxCheck, while measuring RxCheck response time under light and heavy load. The number of request will be progressively increased from low too high to determine the level at which RxCheck backend systems start to falter. During the increase the tester (s) will observe what fails at which point, which can help in identifying if any internal bottlenecks exist that can hamper the operation of RxCheck under different load conditions.

2.2.8 Security Testing

Security testing will focus on two areas

1. Application security which includes access to Database and Data Functions
2. System security which includes login access to server locally or remotely

Application security will ensure that based on user profile (e.g. doctor, nurse, patient), the user will be restricted to specific functions and will have access only to their specific data. This will ensure that only pharmacy users can add patients and only doctors will be able to prescribe medications.

System security will ensure only authorized users will have access to database server and web server through means of authorized access locally or remotely.

2.2.9 Standards Testing

Due to sensitive patient information, this document will focus on testing HIPAA standards and will focus on six areas: Access Control, Encrypted Data Transfer, Data Sanitization, Structured Test Data Approach, Audit Trail and Failover/Load Balancing.

2.2.10 Implementation Testing

The implementation testing plan will begin once the local database has been setup at the end site. This will be a checklist of items, to verify that that certain tasks can be performed. These will consist of creating a generic test patient, assigning two conflicting prescriptions, and verifying that they do show as conflicted on the patient page. This will verify that the local database is working, as well as the connection back to the main database that runs the RXCheck algorithm for conflicts. By performing each of the checks, will ensure that the rest of the system is functioning properly, by nature that each function must be used to complete these tasks.

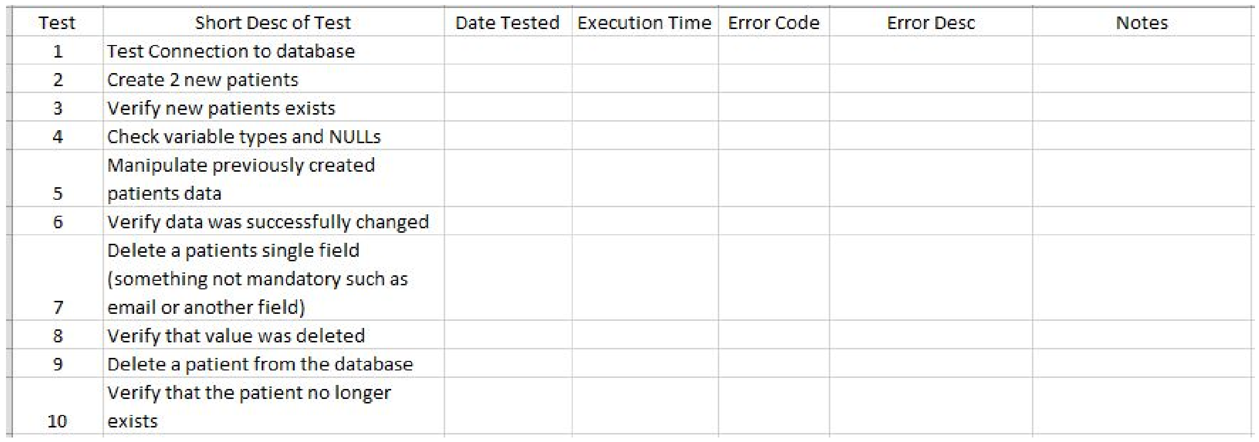
2.2.11 Testing Resources and Staffing

RxCheck will try to have teams of 3 to 4 people each having different parts of the testing cycle, people permitting. Each team will have a test plan that they can run through and if they find an error they make not of it in their test plan. When they are finished with their test plan they will create a ticket under our Footprints system. This is an easy way to keep everything universal and organized as the tickets can be updated and viewable to anyone.

3.0 Testing Procedures

3.1 Test Tracking

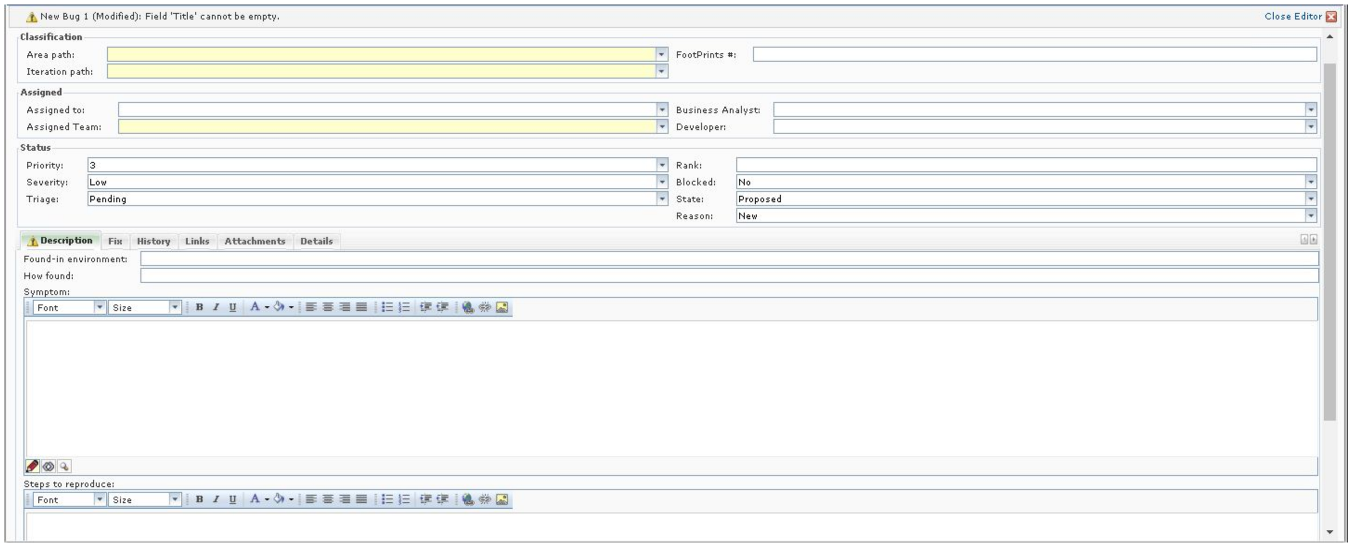
Each team will have a specific test and area to focus on. Each team will receive an Excel spreadsheet that gives them steps A - Z. They are required to run these steps and test outside these steps for erroneous operations within the context of their focus. Below is an example of a database testing spreadsheet.



Test	Short Desc of Test	Date Tested	Execution Time	Error Code	Error Desc	Notes
1	Test Connection to database					
2	Create 2 new patients					
3	Verify new patients exists					
4	Check variable types and NULLs					
5	Manipulate previously created patients data					
6	Verify data was successfully changed					
7	Delete a patients single field (something not mandatory such as email or another field)					
8	Verify that value was deleted					
9	Delete a patient from the database					
10	Verify that the patient no longer exists					

While the spreadsheet is being ran through and filled out, if any errors have occurred, notes should be taken along with steps to replicate the issue. This gives the developers and others a better understanding of what and how the error occurred. Upon completion of the spreadsheet that tester will wait until the following day to report their findings to the team.

These teams will be meeting for quick scrum meetings every day. Each member will report their findings to their team and decide who will input what ticket for these issues. This will help with communication and cut down on duplicate tickets in the system. This should also give a more uniformed and accessible way for everyone to keep up to date.



New Bug 1 (Modified): Field 'Title' cannot be empty.

Close Editor

Classification

Area path:		FootPrints #:	
Iteration path:			

Assigned

Assigned to:		Business Analyst:	
Assigned Team:		Developer:	

Status

Priority:	3	Rank:	
Severity:	Low	Blocked:	No
Triage:	Pending	State:	Proposed
		Reason:	New

Description | Fix | History | Links | Attachments | Details

Found-in environment:

How found:

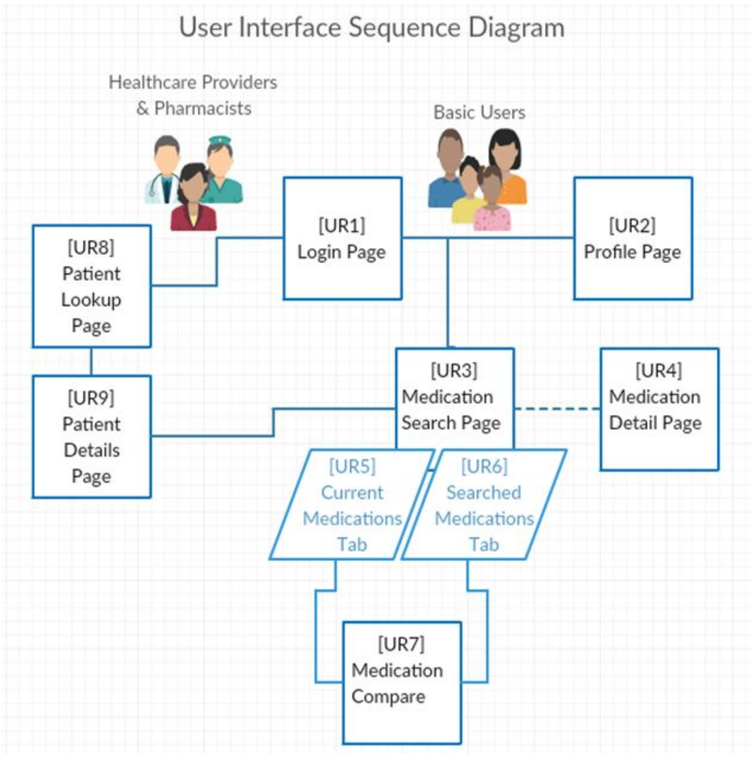
Symptom:

Steps to reproduce:

This is what our tickets will look like as they are going to be input to the system. There are many fields that are required. Some fields such as the FootPrints # will be generated upon submitting this ticket. Below the fields are text boxes for a description of the issue and steps to reproduce. Once submitted an email will be sent to those developing and the team that found the bug. Any update or modification of the ticket will be included in an email update.

3.2 Content Navigation

The navigation testing plan will consist of following the User Interface Sequence Diagram presented below:


User Interface Sequence Diagram
Healthcare Providers & Pharmacists
Basic Users
[UR1] Login Page
[UR2] Profile Page
[UR8] Patient Lookup Page
[UR9] Patient Details Page
[UR3] Medication Search Page
[UR4] Medication Detail Page
[UR5] Current Medications Tab
[UR6] Searched Medications Tab
[UR7] Medication Compare

Log in test:

1. User will be presented with login page when they enter web application address and will enter their credentials to continue
2. Tests will be conducted to ensure proper hyperlinks are working

Profile page:

1. Profile page will be tested to ensure proper presentation of Login Information as well as Personal Information
2. Ensure a message box on top of web app displays a welcome message based on the time of day, a display of the user current logged in and a logout hyperlink is working

Medication Search Page:

1. Tests will be conducted to ensure the search data entry field is available
2. Each row will be tested to ensure they include the Medications, Manufacturer, Dosage Amount and check box columns where user can select medications to compare
3. The Current Medications Details will be tested to ensure Medications, Dosage Amount, Dosage Time and the check box column where user can select medications to compare
4. Ensure a message box on top of web app displays a welcome message based on the time of day, a display of the user current logged in and a logout hyperlink is

Medication Details Page:

1. Test will include clicking on medication which will bring up a medication details overlaid page which includes hyperlinks for Important Info, Before Taking, Dosage, Symptoms, Side Effects
2. Each hyperlink will be clicked upon to ensure proper operation

Current Medications Tab:

1. Tests will be conducted to ensure Medications, Dosage Amount, Dosage Times and a checkbox columns are displayed appropriately with no information missing
2. Users will be able to check and uncheck the boxes to compare medications
3. Ensure a message box on top of web app displays a welcome message based on the time of day, a display of the user current logged in and a logout hyperlink is

Medications Search Tab:

1. This test will include the user clicking on different medication check boxes to ensure this page is displayed
2. Ensure a message box on top of web app displays a welcome message based on the time of day, a display of the user current logged in and a logout hyperlink is

Medication Compare:

1. Tests will be conducted to ensure medications are shown in a grid form
2. Tests will be conducted to ensure conflicting medications are grouped together and conflicting fields will be color coded

3. Ensure a message box on top of web app displays a welcome message based on the time of day, a display of the user current logged in and a logout hyperlink is

Patient Lookup Page:

1. Tests will be conducted to ensure healthcare providers will be able to search patients by name
2. Tests will be conducted to ensure healthcare providers will be able to see results on the same page
3. Tests will be conducted to ensure healthcare providers can click on patients and Patient Details Page loads
4. Ensure a message box on top of web app displays a welcome message based on the time of day, a display of the user current logged in and a logout hyperlink is

Patient Details Page:

1. Tests will be conducted to ensure patient information is displayed correctly in a grid format containing the RecordID, the Record Date, and Facility columns with no missing patient information fields
2. Tests will be conducted to have user click on the History, Current Medications and Dosage log hyperlinks to ensure proper operation
3. Ensure a message box on top of web app displays a welcome message based on the time of day, a display of the user current logged in and a logout hyperlink is

3.3 Database Testing

These test cases should be ran by sending queries to the database or using the webapp under admin rights. If these pass, the database will be fully operational and ready for deployment.

1. Test connection to database inside the web app under admin tools or by queries, if connection was successful the test has been passed.
2. Create new data for the database. Create a patient or two that is under the care of a doctor that is currently in the system.
3. Verify data is now in the database. Check all fields to make sure that they saved properly and that no data was lost.
4. Check NULL types, variable types, and relationships of data and make sure that they are correct. Mismatched data types and fields can cause serious issues later when the database is heavily populated.
5. Manipulate the data in database. Use the following two patients that have been created and change somethings about them.
6. Verify manipulated data saved to database. Check to make sure those fields were the only ones that were changed and changed properly.
7. Delete a piece of data from the database. Delete a piece of data from one patient that can be NULL. Delete the other patient from the database.
8. Verify that the first patient has a NULL value in the proper field. Verify the second patient is no longer a patient in the database or to that doctor.

NOTE: Deleting or manipulating patient data is only for test purposes as this will not be easily done often and by most users.

3.4 User Interface and Component Level Testing

User Interface Testing will be based on the following test cases:

1. Required Fields
   The login page serves as the portal to RxCheck web page and will require the user to enter its username and password. If credentials are not entered, a friendly warning will be generated prompting the user to enter data on the required field.
1. Data Type Errors
   The screen will validate the data that is entered. User can only enter dates, characters, and strings.
1. Onscreen Instructions
   There will be a test to ensure that proper instructions are present on any screen that is not self-explanatory.
1. Progress Bars
   If the screen takes more than 3 seconds to load, it should contain a progress bar so that the user understands the processing is loading.
1. Save and Delete Confirmations
   Profile page, patient details page and new log window allow the user to create, alter, or update data input. The web page should prompt users to save their entry on the current screen prior to moving to another screen. If the user deletes an item, the user will be asked to confirm this action.
1. Type Ahead
   Most of the user interfaces in RxCheck will use drop down list. Typeahead must be enabled to allow the user to skip the items that begins with the first three letters entered.
1. Table Scrolling
   Table scrolling should be allowed for the table information or data that surpass one page.
1. Dialog Box Consistency
   Ensure that the dialog style is consistent on each screen. OK/Cancel dialog should be used throughout the web page.
1. Screen Font Type
   Ensure that the font used from screen to screen is universal. Use of various fonts can be distracting and could be viewed as unprofessional.
1. Menus

The menu items will slightly vary depending on the user's role (basic or elevated user). Ensure that menu items that are not applicable for the specific page are disabled. The order of menu items must be consistent in every screen.

NOTE: As previously noted in database testing, deleting or manipulating patient data is only for test purposes.

Component level testing will start at the debugging phase when developers try to run their solution and project in their IDE. When the product goes into testing phases, developers will create tests that will be ran using Ruby on Rails test environment. Each test will be created for each module, i.e. behind every webpage hitting their controllers.

3.5 Configuration Testing

The configuration test will be conducted by accessing and operating the RxCheck Webapp from different browsers such as Google Chrome, Internet Explorer, Mozilla Firefox, Safari, and Opera; the settings for these browsers will be left as default. The following actions will be performed in each browser to ensure that operation is the same across all testing browsers.

1. Launching the RxCheck Application on different browsers and to observe that pages are rendered correctly.
2. Click all links and observed that they load without problems across all testing browsers.
3. Observe the speed of operation across all browsers and note if performance is negatively or positively affected based on browser.
4. Perform a search on the different browser and observe if Rxcheck is discovered easily by the different search engine on each of the browsers.
5. If errors are found, note the error and submit to ticketing system for tracking and correction.

3.6 Performance Testing

The performance test is going to be focused on the response time and performed by loading the application with computer generated users to observe the performance of the RxCheck under varied load conditions.

1. With the number of users expected to use the app at one time, the tester will perform various operations on the app and observe the response time.
2. After establishing a threshold, tester will incrementally increase the number of users utilizing the app noting the level at which the app starts to slow down and or becomes unresponsive.
3. With the threshold established and results from step 2. Tester will pass information to development team via the ticketing system for error tracking and correction.

3.7 Implementation Testing

The following tests will be conducted to ensure proper implementation. It is assumed the local MSI package to install and configure the Database and the MSI package to install and configure the Webserver are already in place.

1. Multiple database connections will be started to ensure database server is running
2. Multiple web browsers will be directed to the Webserver to ensure the Webserver is running
3. Multiple test patients will be created to test interface between web browser and database server connections are working

4. Assign conflicting prescriptions to a test patient
5. Verify conflicted information shows as defined under the content navigation
6. Heavy user load of 100% utilization will be tested to ensure proper response from webserver and database server

3.8 Security Testing

Application security testing will include the following tests:

1. Verify user profiles to ensure users have access only to authorized information
2. Verify users' profiles to ensure users have access only to functions authorized by their profiles

Security systems testing will include the following tests:

1. Verify only authorized users can log into Web servers and Database servers

The following techniques will be employed to test Application and System security:

1. Identify user profile type and functions available to user (e.g. nurse, patient, pharmacy) and identify which data they have access for
2. Transactions specific tests will be created to ensure users are able only to modify data which is accessible through their profile
3. Users will have permissions modified and transaction specific tests will be created to ensure users are able only to modify data which is accessible through their new profile

Special Considerations: Penetration testing of application server and database will not be covered in this testing plan. This function will be outsourced to security company.

3.9 Standards Testing

Access Control:

1. An access control list allowing users access only to specific applications will be implemented
2. Users will require two factor authentications to log in
3. Users access will be based on their roles

Encrypted Data Transfer:

1. All data will be fully encrypted and only available to users based on their roles

Data Sanitization:

1. During test phase, user specific data will be substituted with generic data

Structured Test Data Approach:

1. Test data will be standardized for verification and validation of specific modules in the application

Audit Trail:

1. All user interactions will be logged
2. Tests will be run to ensure users are not able to modify log data

Failover/Load Balancing:

1. Tests will be run to simulate a server crash and to ensure backup servers will be able to take over with no data loss

Bibliography

"Web Testing: Complete guide on testing web applications." Software Testing Help. N.p., n.d. Web. 21 Apr. 2017.

"10 Checkpoints For Initial User Interface Testing Of Commercial Web Applications." Usability Geek. N.p., 23 Feb. 2015. Web. 21 Apr. 2017.

"Guides.rubyonrails.org." A Guide to Testing Rails Applications - Ruby on Rails Guides. N.p., n.d. Web. 21 Apr. 2017.

"Pragmatic UI Tests." Embedded Star. N.p., n.d. Web. 21 Apr. 2017.

Miller, S. (2009, August 23). 35 Test Cases for User Interface Testing. Retrieved April 21, 2017, from http://www.embeddedstar.com/weblog/2009/08/23/pragmatic-ui-tests/

McCaffrey, J. (2009, January 31). Configuration Testing. Retrieved April 21, 2017, from https://jamesmccaffrey.wordpress.com/2009/01/31/configuration-testing/

Market Share Reports (Source of Analytics Data). (n.d.). Retrieved April 21, 2017, from http://www.netmarketshare.com/