Test Plan Document

For

Nursing Checklist

1 Introduction 3

2 Business Background 3

3 Test Objectives 4

4 Scope 4

5 Test types Identified 5

6 Problems Perceived 5

7 Architecture 5

8 Environment 6

9 Assumptions 6

10 Functionality 6

11 Security 7

12 Performance 9

13 Usability 9

14 Compatibility 11

15 Test Team Organization 11

16 Schedule 12

17 Defects Classification Mechanism 12

18 Configuration Management 14

19 Release Criteria 14

Test Plan

# Introduction

The IUS Nursing Checklist web site is a page that students and faculty/staff use to track student’s task along the time in the program. Previously, students were required to carry a 10 page document with 160+ tasks that they must complete over either 4 semesters or as long as it takes. This means, they have to keep track of one physical document for multiple years. Upon successful return of the document, the Nursing staff then enters each students data into a excel document. This is highly inefficient and cumbersome for both administrators and students alike.

# Business Background

This website streamlines an issue that the IUS administrators of the Nursing department have been facing for years, tracking student progress. Many times, students would lose their document and have to check off another document again to the best of their knowledge. This represents inconsistency and vulnerability in accurate and maintainable data.

# Test Objectives

Testing should assure that our user interface, data layer, and signup/login interactions are working correctly. For user interface, the application should look and feel streamlined and not outdated with current sites. It should be accessible via any means that is possible. The data layer should be encrypted, and safe from student retrieving information as well as malicious attacks by any party. Only administrators should be allowed to access the data layer. Signup and login flows should do checks in the data layer. Signup should encrypt passwords and login should use that same encryption for authorization. Both Signup and Login will contain authorization checks on the data layer to see if users exist or not.

# Scope

User interface should flow well without any problems loading or with button/navigation clicks. It should only display what the controllers tell it to in the code and anything under the hood should not be retrievable in the user interface layer (I’m looking at you JSON blocks). Controllers should designate which information is seen by the UI, by way of database retrieval. The data layer should hold all pertinent information for users and not be retrievable by anyone but that user or an administrator. Passwords or any other sensitive data should be encrypted and not accessible by anyone, not even the admin.

# Test types Identified

This project is not complicated and should suffice from solely manual testing. If there were more complicated queries or different role tiers beyond what is currently in our design, then we may incorporate some sort of testing. As our project is Javascript/Html, Jasmine is great for testing suite for pure Javascript applications such as ours. We may incorporate tests from Jasmine to test Signup/Login features.

# Problems Perceived

Again, this project is very straightforward in its design and has minimal features (as requested by our sponsors). Thus, manual testing has been done every step of the way and the logic is sound. We do not see any perceived problems from design to oracle.

# Architecture

Our Vue Javascript project is very simple in its Architecture. A firebase init is declared, which “hooks up” the database to our Vue project. We then use data access requests in the Javascript to retrieve records and pertinent information for our users and updating our checklist. The other portion is simple authorization in the form of database checks based on user input.

# Environment

The project runs on any device that can connect to the internet and use a web browser. Our website scales for mobile and web alike.

# Assumptions

Signup should check to see if the user is unique, if so we create the user. Login should authorize a user if they are in our database and the correct information is entered; they will be rejected if not. The application must load unique forms for each individual user.

# Functionality

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Responsibility** |
| Constraint 1 | Web accessible device required | Responsibility of user |
| Constraint 2 | Internet required | Responsibility of user |
| Constraint 3 | Must be student/admin of IUS. | Responsibility of user |

***Risk Identified & Mitigation Planned***

Currently there is not a way for user the create a new password if they forget theirs. This is an issue because we cannot guarantee their safe template without this functionality.

***Test Strategy***

We plan to manually test the issues with mock users and data.

***Automation Plans***

Eventually, we wish to implement Jasmine automated testing for our Javascript code.

***Deliverables***

The website will incorporate a good user experience in the form of tracking student progress and will be retrievable by admins.

# Security

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 | Users will have to have their data managed by Firebase (Google). | An albeit reliable, but third party database is being used. |
| Constraint 2 | Public Webpage | Although the web page is an HTTPS, it is still on the web. |

***Risk Identified & Mitigation Planned***

Vulnerabilities exist because we use third party resources such as Firebase, a google cloud database service. Also, the website is publicly hosted on the internet, where all sorts of malicious attacks can occur. This is something out of our control and the users control.

***Test Strategy***

We plan to manually test the issues with mock users and data.

***Automation Plans***

Eventually, we wish to implement Jasmine automated testing for our Javascript code.

***Deliverables***

A functional website hosted online.

# Performance

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 | Website should authorize data. | Web designers should be liable for data unauthorized |
| Constraint 1 | Webpages should respond appropriately | Web designers should be liable unless related to servers |

***Risk Identified & Mitigation Planned***

No risks have been identified on the performance of our application. It is lightweight and efficient.

***Test Strategy***

We plan to manually test the issues with mock users and data.

***Automation Plans***

Eventually, we wish to implement Jasmine automated testing for our Javascript code.

***Deliverables***

The project is hosted on firebase and accessible from a secure HTTPS public url. The website will incorporate a good user experience in the form of tracking student progress and will be retrievable by admins.

# Usability

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 | Website should authorize data. | Web designers should be liable for data unauthorized |
| Constraint 1 | Webpages should respond appropriately | Web designers should be liable unless related to servers |

***Risk Identified & Mitigation Planned***

No risks have been identified on the Usability of our application. It is responsive and easy to follow for all users.

***Test Strategy***

We plan to manually test the issues with mock users and data.

***Automation Plans***

Eventually, we wish to implement Jasmine automated testing for our Javascript code.

***Deliverables***

The project is hosted on firebase and accessible from a secure HTTPS public url. The website will incorporate a good user experience in the form of tracking student progress and will be retrievable by admins.

***14 Compatibility***

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 | Should work on any device that has web browser access | Responsibility of user |

***Risk Identified & Mitigation Planned***

Compatibility with our project is easy. The user just needs a device with internet and a web browser.

***Test Strategy***

We plan to manually test the issues with mock users and data.

***Automation Plans***

Eventually, we wish to implement Jasmine automated testing for our Javascript code.

***Deliverables***

A website fully accessible by anyone with internet access.

# Test Team Organization

The team will manual test all aspect together to assure accuracy. Tests will be done at the time of implementing new functionality as well as at the time of a future functionality. We are all responsible for testing and designing as a team.

# Schedule

The team will meet weekly and have purposeful “check-ins” at specific times. Otherwise, we will randomly communicate and work together to meet specific goals as follows. A fully functional authorization of signup/login. Unique pages loading for each user. Full website functionality and then nice to haves at the end. Video demo for sponsors near the end of the semester.

# Defects Classification Mechanism

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type of Defects | Functionality | Performance | Security | Usability | Compatibility |
| Critical | Works perfect | Peak | Encrypted | Maximum | All browsers |
| Major | Page is functional | Slight delays | Abstracted | Loading | Most browsers |
| Minor | Mostly function | Lag time | Present | Long loading | Select browsers |
| Cosmetics | Parts of UI loading | Slow | Public | Not so much | Edge only |

***Defects Logging and Status Changing Mechanism***

All defect and logging are sent to either Github (version control) and/or Firebase firestore.

***Turn Around Time for defect fixes***

No service has been negotiated for defects. Will work for $35 an hour.

# Configuration Management

We are using npm node modules and javascript html. Any text editor can use these architectures/frameworks. We chose Visual Studio Code since its extremely lightweight and responsive. It also have many useful programming tools to become a robust IDE; it also has the Spotify app.

# Release Criteria

The website will be release and hosted on firebase under their “free” tier where it has restrictions on the amount of saved data, daily reads/writes, etc. We don’t anticipate this page every going beyond those free limits; however, it is highly upgradable for a paid plan and fairly cheap in the long run if it did pick up traction.