**Software Requirements Specification**

**Purpose:** To create a web/mobile environment that allows students to track progress and staff/faculty to monitor their progress in the Nursing Program.

**System Overview**

The product will be a webform that has a set of static information for users to manipulate. We will be using Vue.js for all tiers of the application. Vue has well documented adaptation with Firebase as a no-SQL database option but the potential to use SQL instead is also realistic. We will use Jasmine for unit testing.

**Overall Description**

**System Interfaces**

We will use Vue.js as a controller for our database. The best documented option for Vue.js at this point is Firebase.

**User Interfaces**

There will be two separate interfaces, a “Student UI” and a “Admin UI.” The student interface will have a list of items that they can manipulate while the admin interface will allow querying of student progress.

**Hardware Interfaces**

Computers are our main target since the list of items is very large; however, we know realistically that many people will wish to access this webform via their mobile device on a mobile web browser at the very least. We plan to integrate interfaces scalable for Android and IOS devices as well.

**Software Interfaces**

For the student interface, the user will send a status of their checklist item to the database. Since the sole purpose of the interface is to update items, the user does not need much in the way of retrieval. However, the admin interface will be retrieving information from the database. The user will be sending queries and retrieving data.

**Memory Constraints**

This product will be minimalistic and have a small amount of traffic. We do not foresee any kind of constraint on memory being an issue.

**Adaptation Requirements**

The product needs to be needs to be easy to use for any person, regardless of their experience with technology. It needs to be accessible via computer web browser and mobile web browser.

**Product Functions**

* Sign-up/Login interface populates the correct user interface
* Student UI allows for quick and easy updating of their checklist items
* Student UI shows progress of items
* Admin UI allows for quick and easy querying of student data
* Admin UI shows returns data from queries

**User Characteristics**

There will be two types of users, the student and the admin. The student user should be anyone designated by the admin as “Student.” The admin will be designated from sign-up/log-in from a check in the database. Neither user needs special training to perform the requirements of the product.

**Constraints**

The product needs to be simple to use since we will have users of all technological skill levels.

**Assumptions**

Admins will accurately properly input student user’s information.

**Dependencies**

**Specific Requirements**

**External Interface Requirements**

Site flow:

Sign up/login

If Admin, go to 2. Else, go to 3.

1. Admin UI (Ability to view progress of students/query data)
2. Student UI (ability to view/manage tasks)

**Functional Requirements**

The product will need to be able to initially query the database to determine whether a student or an admin are trying to sign up/login. In the Student UI, the product will take in user input in the form of updating tasks and send them to the database. In the Admin UI, the product will take in user input in the form of querying data. Resulting output/text will be displayed depending on the task.

**Performance Requirements**

The product will need to be accessible on web and mobile alike. There will be no issue with the web version, but the mobile version will need to be adequately fast for the users to find it usable.

**Design Constraints**

There is very specific data that needs to be encapsulated and we must follow the guidelines given to us with our static information.

**Standards Compliance**

Any borrowed services, whether it be code, images, or other tools will be open source or creative commons licenses.

**Logical Database Requirement**

Any database will work as we will not need many tables to completely meet our complexities.

**Security**

We will need to keep user information secure using popular techniques in the industry (to be determined)

**Maintainability**

Ideally, our product will not require maintenance and any tweaks that need to happen will be made accessible on the Admin UI.