Software Requirements Specification

**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.

**Subsystem Decomposition**

**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.

**Data Interfaces**

We will use Vue.js as a controller for our database. The best documented option for Vue.js at this point is Firebase. Vue is responsible for bringing in a list of student tasks as well as updating/managing them with Firebase.

**Hardware/Software Mapping**

**Hardware**

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**

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.

**Persistent Data Management**

**Data Design**

We will have a Firebase database to manage persistent data. Anything that is changed via the Student UI or Admin UI will be persistent between sessions and stored in the database. Firebase does not use Schemas and Tables like SQL does. Instead, Firebase is just a big JSON object where you can store a combination of key/value pairs. The thought process is similar to SQL but it looks completely different. We will need a “taskStore” key that we can store/retrieve our list from. Also, a “taskAdd” that adds a new task to “taskStore.” Perhaps admins will get privileges to a “taskInactive” key that will remove an item from “taskStore” and hold it in “taskInactive.” For user varication we will need a “users” key to store users and have a check to verify whether a user is a student or admin. Further, we will need a “addStudent” key and “addAdmin” key that puts a student or admin in “users.” There are other optimizations for the database but this is the baseline.

**Access Control and Secturity**

**Authentication**Upon sign-up/login, there will be a check with the database to see whether the user is a student/admin and populate the correct UI accordingly. The Student UI allows for quick and easy updating of their checklist items and shows their current state. The Admin UI allows for quick and easy querying of student data and returns data from queries.

**Global Software Control**

**Event Driven**The application will be event-driven. Events that will be triggered are changing task completion, adding tasks, and querying data. UI’s will update in real-time.

**Boundary Conditions**

**System States**

One concern of a website is how data is stored in session and out of session. Simply navigating to the website will start the application and any data altered in session will update and save in real time. Firebase is a real-time cloud database so data is synced and remains available offline. There are no anticipated efforts needed to upkeep, even if hardware is presented. Firebase is a google product and also has webhosting (owned by google) that will be stable for the foreseeable future.

**Contributions**

This project does not require much in the way of split declarations to reach the final product. The contributions of each team member will be in the full stack and we will be working on version control with GitHub, mainly to stray from redundancy/overlap. This is intended not only because of the straight-forward design but also for our own knowledge of the full process.