**CoolScan**

Nonfunctional Requirements Definition

Release is a number for the document being released for review. A release can go through many changes between the versions. That’s why the SaveDate field below is important – to distinguish between Versions of a Release.

Kevin Chan, Jesse Hernandez, Daniel Jacuinde, Taranveer Kaur

This template developed by Requirements Quest® is based on IEEE (Institute of Electrical and Electronics Engineers) 830 Standard SRS (Software Requirements Specification). For additional information, visit <http://www.ieee.org>. To request an electronic copy of this template, send an email to [inquire@requirementsquest.com](mailto:inquire@requirementsquest.com) or visit [www.RequirementsQuest.com](http://www.RequirementsQuest.com).



Table of Contents

[1. Introduction 3](#_Toc26388359)

[1.1. Purpose of this document 3](#_Toc26388360)

[1.2. Reference Materials 3](#_Toc26388361)

[1.3. Assumptions 3](#_Toc26388362)

[1.4. Constraints 3](#_Toc26388363)

[1.5. Dependencies 3](#_Toc26388364)

[2. Nonfunctional Requirements 4](#_Toc26388365)

[2.1. Nonfunctional Requirement Statements 4](#_Toc26388366)

[3. Appendices 6](#_Toc26388367)

[3.1. Requirements Issues 6](#_Toc26388368)

# Introduction

## Purpose of this document

This document contains all nonfunctional requirements for this project.

This document must establish an agreement between the customer and suppliers on what the system must do. The contents of this document should endeavor to reduce development effort by reducing redesign due to ambiguous statements. It should contain enough detail to provide a basis for estimating cost and schedules. Each version should provide a baseline for verification and validation. It should facilitate “transfer” of knowledge, commitment, intent and finally transfer of the viable product to the stakeholders. The final version of this document should accurately and completely describe the final product and serve as a basis for future enhancements.

Both the writer and the reader should strive for these goals in the elicitation, analysis, presentation and validation of this document. Through baselining, versions of this document should clearly identify all changes.

## Reference Materials

There are many other documents that together describe the complete set of requirements for this project.

Other documents such as standards, regulations, business process flows, or descriptions of existing functionality to be modified should be named here. Often there are copies of common information distributed throughout the company. The definitive source of the information and/or the person who manages the information should be noted so that discrepancies may be quickly resolved with clear authority.

| **Reference Document Name** | **Brief Description** | **Location of Definitive Source** |
| --- | --- | --- |
| Project Wiki | Explanation of project and specifications | GitHub Repository |

## Assumptions

Identify anything that adds clarification to or provides background information about the nonfunctional requirement statement, or other related item.

Assumption statements are not specific requirements themselves. They help clarify requirements that may be misinterpreted due to different definitions of terms or different opinions about how some business operation flows. During validation of requirements, assumptions may help discover omitted requirements, highlight project risks, or encourage discussion of various interpretations. Examples include: technology such as a specific operating system or database; and expectations of the development team.

| **ID** | **Assumption Statement** | **Related To** |
| --- | --- | --- |
| A1 | All students have been given credentials from admit | Log in |

## Constraints

Identify anything that puts limits on implementing the nonfunctional requirements.

This section is often misunderstood. Pay particular attention to this description. A constraint is a statement that expresses measurable bounds on the implementation of a system. It may limit the technology or environment in which the system may be built. It may list the certain options that may be considered for implementation. It may describe limitations or conditions that users must work under. In all cases, a “constraint” is differentiated from business, user and functional requirements in that it does not describe the functionality but rather limits the design or implementation options. Examples of constraints include: hardware constraints because the software may reside on a server with other applications and compete for resources such as memory, ports, and processor speed; reliability constraints such as how often the system must be backed up or even if hot back such as RAID is required; and “criticality” refers to how long the system would be allowed to be out of operation due to some failure. These are just a few of the constraints that may make the project a success.

| **ID** | **Constraint Statement** | **Related To** |
| --- | --- | --- |
| C1 | Security, the verification and encryption are still not complete | Safety |

## Dependencies

Detail any external event, condition, or system that must be in place for a requirement to be implemented.

| **ID** | **Dependency Statement** | **Related To** |
| --- | --- | --- |
| D1 | Must have internet access, connected to server. | Access |

# Nonfunctional Requirements

## Nonfunctional Requirement Statements

Column Header Key:   
BR = Business Rules Identifier, CI = Common Information Identifier, S = Status, P = Priority

| **ID** | **Nonfunctional Requirement Statements** | **BR** | **CI** | **S** | **P** |
| --- | --- | --- | --- | --- | --- |
| **OPERATION Requirements**: How well does the system perform for daily use?  Describe the user concern for using the functionality. The user perceives the system as a tool to automate tasks. | | | | | |
| **Access Security** How well is the system guarded against unauthorized access? The extent to which the system is safeguarded against deliberate and intrusive faults from internal and external sources. | | | | | |
| N-ACS1 | The users are only able to register with the admin provided ID. |  |  |  |  |
| **Accessibility** How easy is the system to use by people with varying capabilities? The extent to which the software system can be used by people with the widest range of capabilities to achieve a specified goal in a specified context of use. | | | | | |
| N-ACC1 | Each button has written out words to its function. Icons are large and clear. Help button is located on an easy to find spot. |  |  |  |  |
| **Availability** How dependable is the system during normal operating times? The degree to which users can depend on the system to be up (able to function) during “normal operating times.” | | | | | |
| N-AVL1 | The system is highly dependable on the central database. The check in, planner and grades are completely base on it. |  |  |  |  |
| **Confidentiality** How well does the system make sensitive data available to authorized users? The degree to which the software system protects sensitive data and allows only authorized access to the data. | | | | | |
| N-CNF1 | App tracks students’ locations, grades, plans, and conversations. All data is saved in a secure database. Teachers can only view grades. Students can only view their own data. |  |  |  |  |
| **Integrity** How accurate and authentic are the data? The degree to which the data maintained by the software system are accurate, authentic, and without corruption. | | | | | |
| N-INT1 | Check in data is tracked by the GPS, the inputs are checked with student schedules and location of class to validate the entry.  The planner events are private to the student, so it does not impact anything what they save.  Grades are view only for student and can only be changed by teacher and overseen by school admin.  The inbox is linked to specific students depending on their id. Only that student can access the chat and send the message. |  |  |  |  |
| **Reliability** How immune is the system to failure? The extent to which the software system consistently performs the specified functions without failure. | | | | | |
| N-REL1 | The code has inter-dependencies with the database. If connection is breaking the system will fail. The school is asked to keep proper data in organized fashion to avoid this. |  |  |  |  |
| **Safety** How well does the system prevent harm to people and the environment? The degree to which a software system prevents harm to people or damage to the environment in the intended context of use. | | | | | |
| N-SAF1 | The application does not have any reach to the environment. Students do take their scores seriously, and that’s why the teacher is the only one allowed to alter them and administration has the power to view and flag if the data is missing. All this to prevent emotional harm for the students. |  |  |  |  |
| **Survivability** How resilient is the system from failure? The extent to which the software system continues to function and recovers in the presence of a system failure. | | | | | |
| N-SRV1 | System has frequent error catching messages. In case of failure, the user will be informed and redirected to the last working page. |  |  |  |  |
| **Usability** How easy is it to learn and operate the system? The ease with which the user is able to learn, operate, prepare inputs, and interpret outputs through interaction with a system. | | | | | |
| N-USE1 | The system is very intuitive. Each function has a well labeled button and each screen has a return to home button. All pages are titled to let the user know where they are. |  |  |  |  |
| **REVISION Requirements**: How easy is it to correct errors and add functions?  Describe the user concern for changing source code or data that drive the system. The user perceives the system as programmed language statements. | | | | | |
| **Flexibility** How easy is it to modify to work in different environments? The ease with which the software can be modified to adapt to different environments, configurations, and user expectations. | | | | | |
| N-FLX1 | It is possible to transfer the work on a different environment. The only difficulty will be changing the html file. The change will be simple since the Ionic framework uses very adaptable code. |  |  |  |  |
| **Maintainability** How easy is it to upkeep and repair the system? The ease with which faults in a software system can be found and fixed. | | | | | |
| N-MNT1 | Fairly easy, every task is built up from individual functions. The functions are clearly labeled. |  |  |  |  |
| **Modifiability** How easy is it to change the software system, and at what cost?  The degree to which changes to a software system can be developed and deployed efficiently and cost effectively. | | | | | |
| N-MDF1 | Not easy. Changing the system will require significant work and will cost. |  |  |  |  |
| **Scalability** How easy is it to expand or upgrade the system’s capabilities? The degree in which the system is able to expand its processing capabilities upward and outward to support business growth. | | | | | |
| N-SCL1 | Easy. The system components are kept separate and more features can be developed within them. |  |  |  |  |
| **Verifiability** How easy is it to show the system performs its functions? The extent to which tests, analysis, and demonstrations are needed to prove that the system will function as intended. | | | | | |
| N-VER1 | The system requires students to use it and with dummy entries in the database, it can be easily verified. |  |  |  |  |

# Appendices

## Requirements Issues

| **ID** | **Description** | **Raised by** | **Assigned to** | **Status** |
| --- | --- | --- | --- | --- |
| IS1 | Database malfunction | Team 9 | Kevin | IP |
| IS2 | GPS malfunction | Team 9 | Jesse | IP |
| IS3 | Single user using multiple logins for fake check-in | Team 9 | Tania | IP |
| IS4 | Location spoofing | Team 9 | Daniel | IP |