# Especificación de Requisitos de Software (SRS)

## Proyecto: Calculadora de Nota Faltante

Versión: 1.0

Autores: Lina Franco Martínez

Fecha: 29 de octubre de 2025

## 1. Introducción

### 1.1 Propósito

El propósito de este documento es definir y especificar todos los requisitos para el sistema "Calculadora de Nota Faltante". Este sistema es una aplicación web *client-side* diseñada para calcular la nota mínima requerida en el tercer corte de una asignatura para alcanzar una nota final de aprobación (3.0), basándose en las notas de los dos primeros cortes y sus ponderaciones fijas.

Este SRS sirve como el contrato fundamental entre los *stakeholders* y el equipo de desarrollo, detallando qué debe hacer el sistema.

### 1.2 Alcance del Producto

El producto es una herramienta de software independiente que se ejecuta en el navegador web del usuario. El alcance del sistema se limita a:

* Aceptar la entrada de dos notas numéricas (Corte 1 y Corte 2).
* Validar que las entradas sean numéricamente válidas y estén dentro del rango académico (0.0 a 5.0).
* Procesar estas entradas usando una fórmula de cálculo predefinida.
* Presentar al usuario uno de tres posibles escenarios:
  1. La nota exacta necesaria en el tercer corte.
  2. Un mensaje indicando que la materia ya está aprobada.
  3. Un mensaje indicando que es matemáticamente imposible aprobar.

El sistema **no** almacenará datos, no requerirá autenticación de usuarios y no se integrará con ningún sistema de gestión académica.

### 1.3 Definiciones, Acrónimos y Abreviaturas

|  |  |
| --- | --- |
| **Término** | **Definición** |
| **SRS** | Especificación de Requisitos de Software. |
| **UI** | Interfaz de Usuario (User Interface). |
| **DOM** | Document Object Model (Modelo de Objetos del Documento). |
| **CDN** | Content Delivery Network (Red de Distribución de Contenido). |
| **NaN** | Not a Number (No es un Número). |
| **Corte** | Período de evaluación académica. |
| **Client-Side** | Lógica que se ejecuta en el navegador del usuario. |

### 1.4 Referencias

Este documento se basa en el análisis de los siguientes artefactos de código:

1. Documento de Luis/parcial/index.html (Estructura de la Interfaz)
2. Documento de Luis/parcial/styles.css (Estilos visuales personalizados)
3. Documento de Luis/parcial/script.js (Lógica de negocio y validaciones)
4. Documento de Luis/CODIGO DE LINA FRANCO Y LUIS GALARAGA.txt (Documentación explicativa)

### 1.5 Resumen del Documento

Este SRS está organizado en tres secciones principales:

* **Sección 1 (Introducción):** Define el propósito, alcance y contexto del documento.
* **Sección 2 (Descripción General):** Proporciona una visión de alto nivel del producto, sus usuarios y sus restricciones.
* **Sección 3 (Requisitos Específicos):** Detalla todos los requisitos funcionales, no funcionales y de interfaz del sistema.

## 2. Descripción General

### 2.1 Perspectiva del Producto

El sistema es una aplicación web *single-page* (SPA) de propósito único. Es un producto nuevo e independiente. No reemplaza ningún sistema existente y opera de forma autónoma, con la única dependencia externa de la CDN de Tailwind CSS para el renderizado de la interfaz.

### 2.2 Funciones del Producto

La funcionalidad principal del sistema se resume en la función calcularNota(), que:

1. Recopila y valida las entradas del usuario.
2. Limpia cualquier estado de resultado anterior.
3. Calcula la nota necesaria.
4. Muestra el resultado con el formato y estilo adecuados.

### 2.3 Características de los Usuarios

El único tipo de usuario previsto es el **"Estudiante"**.

* **Conocimiento:** Se asume que el usuario tiene conocimientos básicos de navegación web (ingresar texto en campos, hacer clic en botones).
* **Motivación:** El usuario está motivado para determinar rápidamente su situación académica frente al último corte de una asignatura.

### 2.4 Restricciones Generales

* **R-01:** El sistema debe funcionar exclusivamente en el lado del cliente (navegador). No se permite lógica de *backend*.
* **R-02:** La nota mínima de aprobación está fijada (hardcoded) en **3.0**.
* **R-03:** Las ponderaciones de los cortes están fijadas: Corte 1 (33%), Corte 2 (33%) y Corte 3 (34%).
* **R-04:** El sistema debe usar la CDN de Tailwind CSS para sus estilos base.
* **R-05:** El sistema debe aceptar tanto el punto (.) como la coma (,) como separadores decimales para mejorar la usabilidad.

### 2.5 Suposiciones y Dependencias

* **S-01:** El usuario accede a la aplicación desde un navegador web moderno que soporta JavaScript (ES6).
* **D-01:** Se requiere una conexión a Internet activa en la primera carga para obtener la librería de Tailwind CSS desde https://cdn.tailwindcss.com.

## 3. Requisitos Específicos

### 3.1 Requisitos Funcionales (RF)

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requisito** | **Descripción Detallada** | **Verificado en** |
| **RF-01** | Entrada de Nota Corte 1 | El sistema debe proveer un campo de texto (input#corte1) para que el usuario ingrese la nota del primer corte. | index.html (L:20) |
| **RF-02** | Entrada de Nota Corte 2 | El sistema debe proveer un campo de texto (input#corte2) para que el usuario ingrese la nota del segundo corte. | index.html (L:26) |
| **RF-03** | Normalización de Separador Decimal | El sistema debe reemplazar la coma (,) por el punto (.) en las entradas del usuario antes de la conversión a float. | script.js (L:60, L:61) |
| **RF-04** | Activación del Cálculo | El sistema debe ejecutar la función calcularNota() cuando el usuario haga clic en el botón "Calcular Nota Necesaria". | index.html (L:32) |
| **RF-05** | Validación de Campos Vacíos | El sistema debe verificar que ambos campos de entrada contengan un valor. Si uno está vacío, debe mostrar un mensaje de error. | script.js (L:63-L:67) |
| **RF-06** | Validación de Formato Numérico (NaN) | Tras la normalización, si el valor no puede ser interpretado como un número (isNaN), el sistema debe mostrar un error. | script.js (L:73-L:77) |
| **RF-07** | Validación de Rango de Entrada (0-5) | El sistema debe verificar que las notas ingresadas estén en el rango [0, 5]. Si no lo están, debe mostrar un error. | script.js (L:80-L:84) |
| **RF-08** | Lógica de Cálculo | El sistema debe aplicar la fórmula: | $$Nota\_3 = \frac{3.0 - (C\_1 \cdot 0.33) - (C\_2 \cdot 0.33)}{0.34}$$ | script.js (L:88) |
| **RF-09** | Presentación de Resultado: Imposible | Si Nota\_3 > 5.0, el sistema debe informar al usuario que es imposible aprobar y mostrar la nota requerida (formateada a 2 decimales). | script.js (L:91-L:93) |
| **RF-10** | Presentación de Resultado: Aprobado | Si Nota\_3 <= 0.0, el sistema debe informar al usuario que la materia ya está aprobada. | script.js (L:96-L:98) |
| **RF-11** | Presentación de Resultado: Necesario | Si 0.0 < Nota\_3 <= 5.0, el sistema debe informar al usuario la nota exacta que necesita (formateada a 2 decimales). | script.js (L:99-L:101) |
| **RF-12** | Limpieza de Estilos Previos | Antes de mostrar un nuevo resultado, el sistema debe eliminar las clases de estado (info, success, error) del div#resultado. | script.js (L:54-L:56) |
| **RF-13** | Visualización del Resultado | El sistema debe hacer visible el contenedor div#resultado (quitando la clase hidden) al ejecutar un cálculo. | script.js (L:49) |

### 3.2 Requisitos de Interfaz Externa

#### 3.2.1 Interfaz de Usuario (UI)

* **UI-01:** La interfaz debe ser limpia, centrada y responsive, adaptándose a dispositivos móviles (ver meta name="viewport" y clases de Tailwind).
* **UI-02:** La paleta de colores para la retroalimentación de resultados debe ser:
  + **Informativo (Caso Necesario):** Fondo azul, texto azul oscuro (Clase .info).
  + **Éxito (Caso Aprobado):** Fondo verde, texto verde oscuro (Clase .success).
  + **Error (Validación o Imposible):** Fondo rojo, texto rojo oscuro (Clase .error).
* **UI-03:** El área de resultados (div#resultado) debe estar oculta (class="hidden") al cargar la página.
* **UI-04:** Los campos de entrada deben tener *placeholders* (placeholder="...") que guíen al usuario sobre el formato de entrada (Ej: "3.5 o 4,2").

#### 3.2.2 Interfaz de Software

* **SWI-01:** El sistema depende de la librería Tailwind CSS, cargada externamente desde https://cdn.tailwindcss.com.
* **SWI-02:** La lógica de la aplicación se encapsula en script.js, que manipula el DOM del index.html.
* **SWI-03:** Los estilos personalizados se cargan desde styles.css.

#### 3.2.3 Interfaz de Hardware

* No existen requisitos específicos de hardware más allá de un dispositivo capaz de ejecutar un navegador web moderno (PC, smartphone, tablet).

#### 3.2.4 Interfaz de Comunicaciones

* La aplicación utiliza el protocolo HTTPS para recuperar la dependencia de Tailwind CSS (SWI-01). No se realizan otras comunicaciones de red.

### 3.3 Requisitos No Funcionales (RNF)

|  |  |  |
| --- | --- | --- |
| **ID** | **Tipo** | **Requisito** |
| **RNF-01** | Rendimiento | Todos los cálculos y validaciones deben completarse de forma instantánea (perceptiblemente < 100ms) al ser operaciones *client-side*. |
| **RNF-02** | Usabilidad | El sistema debe ser autoexplicativo. La retroalimentación de errores debe ser clara y orientar al usuario (ej. "Las notas deben estar entre 0 y 5."). |
| **RNF-03** | Robustez | El sistema debe manejar entradas malformadas (letras, múltiples decimales, rangos incorrectos) sin fallar (crash) y mostrando el error correspondiente (Ver RF-05, RF-06, RF-07). |
| **RNF-04** | Mantenibilidad | El código debe estar claramente separado por responsabilidades: Estructura (.html), Estilo (.css) y Lógica (.js). |
| **RNF-05** | Portabilidad | La aplicación debe funcionar de manera consistente en los principales navegadores web modernos (Chrome, Firefox, Safari, Edge). |

## 4. Apéndice A: Casos de Uso

### 4.1 Caso de Uso 01: Calcular Nota Necesaria (Flujo Exitoso)

* **Actor:** Estudiante
* **Precondición:** El Estudiante ha abierto la aplicación web index.html.
* **Flujo Principal:**
  1. El Estudiante ingresa "4,0" en el campo input#corte1.
  2. El Estudiante ingresa "3.5" en el campo input#corte2.
  3. El Estudiante presiona el botón "Calcular Nota Necesaria".
  4. El Sistema (via calcularNota()) limpia estilos previos (RF-12).
  5. El Sistema valida que los campos no están vacíos (RF-05).
  6. El Sistema normaliza las entradas a "4.0" y "3.5" (RF-03).
  7. El Sistema valida que son números (RF-06) y están en rango (RF-07).
  8. El Sistema calcula: (3.0 - (4.0 \* 0.33) - (3.5 \* 0.33)) / 0.34 = 0.525 / 0.34 = 1.544... (RF-08).
  9. El Sistema determina que el resultado está entre 0 y 5 (RF-11).
  10. El Sistema muestra el div#resultado con el texto "Necesitas sacar **1.54** en el tercer corte..." y aplica la clase info (RNF-02, UI-02).
* **Postcondición:** El Estudiante ve la nota que necesita para aprobar.

### 4.2 Caso de Uso 02: Calcular Nota (Flujo de Error: Imposible)

* **Actor:** Estudiante
* **Precondición:** El Estudiante ha abierto la aplicación web.
* **Flujo Principal:**
  1. El Estudiante ingresa "1.0" en el campo input#corte1.
  2. El Estudiante ingresa "1.0" en el campo input#corte2.
  3. ...
  4. El Sistema calcula: (3.0 - (1.0 \* 0.33) - (1.0 \* 0.33)) / 0.34 = 2.34 / 0.34 = 6.88... (RF-08).
  5. El Sistema determina que el resultado es > 5.0 (RF-09).
  6. El Sistema muestra el div#resultado con el texto "Necesitas sacar **6.88**... ¡Es imposible aprobar! 😥" y aplica la clase error.
* **Postcondición:** El Estudiante es informado de que no puede aprobar.

### 4.3 Caso de Uso 03: Calcular Nota (Flujo de Validación: Rango)

* **Actor:** Estudiante
* **Precondición:** El Estudiante ha abierto la aplicación web.
* **Flujo Principal:**
  1. El Estudiante ingresa "6.0" en el campo input#corte1.
  2. El Estudiante ingresa "3.0" en el campo input#corte2.
  3. El Estudiante presiona el botón.
  4. ...
  5. El Sistema detecta que "6.0" está fuera del rango (RF-07).
  6. El Sistema muestra el div#resultado con el texto "Las notas deben estar entre 0 y 5." y aplica la clase error.
* **Postcondición:** El Estudiante es informado sobre el error en la entrada.