Software Requirements Specification

for

MATH_SCANNER_SOLVER

Version 1.3 approved

Prepared by Jonathan O’Berry, Sarah Baron, and Carlos Gomez

Software Engineer Class UNF

April 25, 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Carlos Garcia Gomez	3/7/22	Started	1.0
Sarah Baron	3/25/22	Document Updates	1.1
Jonathan O'Berry	4/3/22	Writing sections 4-6	1.2
Jonathan O'Berry	4/24/22	Correcting Grammar, and Significant additions to sections 1-3	1.3

1. Introduction

1.1 Purpose

The product whose software requirements are specified in this document is named the “Math Scanner Solver”, and is a whole product. This product will receive an uploaded photo or the product will allow the user to type an equation or will allow the user to take a photo. The software will then recognize a math equation either contained within the photo, or typed in by the user, and the product will solve the problem for the user, outputting the result. The math problems are limited to simple numeric calculations and trigonometry.

1.2 Document Conventions

All text in this SRS document is in the Arial font type. All subtitles for sections of this document are in 23 point sized font, and are bolded. All subtitles for subsections of the sections in this document are in 17 point font and are bolded, and all normal text within this document is in 11 point font. All requirements within this document have their own priority.

1.3 Intended Audience and Reading Suggestions

This document is intended for project managers, developers, and testers to use as a guide for the intended product, along with marketing staff to use as material to sell the product to users, therefore we are providing a realistic specification of the app and its features which will be updated as the product is developed. It is intended to be read in the order it is written. Developers should focus primarily on sections 2.0 and onward. Marketing staff and project managers should focus on section 3.0 and onward.

1.4 Product Scope

The product will provide users such as educators, students and professionals a quick way to solve simple equations, much like a regular calculator, but through the use of image uploads and a camera. This will benefit students or any personnel in solving mathematical equations. It is an easy to use web app where the user can upload the image and have an instant solution for the equation.

1.5 References

N/A

2. Overall Description

2.1 Product Perspective

The product is a new self contained product making use of APIs and Libraries already available. In this, the product is in part making use of the reuse oriented model. This product is intended for public use, and if deployed in a real world setting could even take the form of a publicly available website. This product is similar to already extant products in that it centers around the concept of calculating the answer to an equation by taking a picture of that equation.

2.2 Product Functions

1. The product must be able to properly interpret what type of input the user selected.

2. The product must be able to convert images and typed strings into LaTeX.

3. The product must be able to detect what type of calculation is being attempted.

    - Numeric Calculation (2+2)
    - Trigonometric Calculation

4. The product must be able to calculate the answer to the basic equation entered by the user.

    - This should take a reasonable amount of time (<2s)

5. The product must print the result of the calculation to the user.

2.3 User Classes and Characteristics

There will be no differences between users. All users will have access to the same web application with the option to upload a picture, take a picture, draw an equation, or type an equation. These users will be able to solve simple math problems and trigonometric functions. In future iterations when there are more features a premium account may be created in which users would be charged to solve more complex equations, or advertisements may be implemented for regular users. However as previously stated, in the current version there are no different classes of users, all users fit under one class.

2.4 Operating Environment

The product is a web based application. To run the product the system that the product is running on must have both Node.js and React installed. Additionally the product will make use

of the MathPix API, in order to parse photos into equations that can be calculated. The product also makes use of the MathQuill library. The MathQuill library is used to convert typed strings into LaTeX text that can be used more easily in calculations.

2.5 Design and Implementation Constraints

There are few limitations to the current design of the product. The largest limitation is that the number of requests to interpret photos must be kept to a small amount, as it costs us $0.004 for every photo run through the MathPix API. This means that a cap would have to be put on the number of runs that may be sent to MathPix, and while this does not affect development or testing, this could limit how useful the product would be to users if actually deployed. Another limitation to our program is that since under the current design it is calculating the equation itself, there are many types of equations that it is not able to calculate. Additionally, another limitation is that the user must install both React and Node.js before they can run the program.

2.6 User Documentation

A README file exists within the github containing the product. The product interface is relatively easy to use with icons containing text explanations. There are also several UML diagrams of this product that show how it works from many different perspectives.

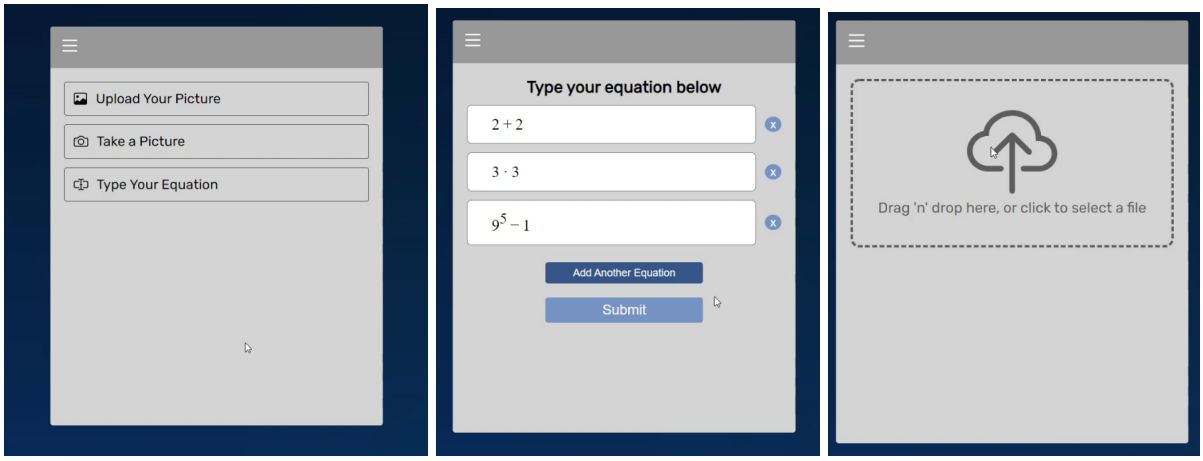
2.7 Assumptions and Dependencies

It is assumed that users will not intentionally upload photos of things that are not mathematical equations. It is also assumed that MathPix and Mathquill will correctly output data based upon the input they receive. The product depends on MathPix correctly parsing images into LaTeX text, and MathQuill correctly converting strings into LaTeX text.

3. External Interface Requirements

3.1 User Interfaces

The product has one main user interface. There are options to type an equation, upload a picture, and take a picture. Multiple equations can be uploaded at once and will be solved simultaneously. Features may be accessed by clicking the button labeled with the name of each feature. Once the icon is selected the user will be brought to a new page which will allow them to upload a picture, take a picture, or type an equation. Through the use of React this does not actually switch to different pages, instead modifying the displayed page, and keeping the same relative format. A user interface is needed for basically all software components of this product as this product centers around the idea of the user entering in an equation and being given an answer.



3.2 Hardware Interfaces

When the user begins using the product, they will use the mouse to select the option representing in what manner they intend to input the equation. If the user selects the camera option, then the product will request permission to access the system camera. The view from the system camera is then displayed by the product, and the user uses the mouse to click on the button once their equation is within view of the camera. The camera will then capture what is currently in view. If the user selects the upload image option,they will be redirected to a page where they will use their mouse to click a button to upload a photo and then use the mouse to select what photo they are uploading, or the user can use the mouse to drag the photo they wish to upload ot the box. If the user selects the option to type an equation they will use the mouse to click on the text box, and then use the keyboard to type their input.

3.3 Software Interfaces

The product will send images that the user inputs to the MathPix api. The MathPix api will return to the product a json file containing the equation it parsed in LaTeX form. If the user chooses to type in an equation, the string is converted into LaTex form using the MathQuill library. This is all completed on the front-end primarily using the React Library of JavaScript. From this point the information is sent to the back end which uses the Node.js runtime environment. The back end now takes this LaTeX equation and calculates the answer and returns that answer to the front end which displays the answer to the user. This product also makes use of the react-dropzone and react-webcam components.

3.4 Communications Interfaces

The product can be used through any Chromium web browser, or the safari web browser. The product may be able to be used through other web browsers, however proper functioning and formatting is not guaranteed. The product communicates with APIs that parse handwriting from images using the HTTP standard. Other than this there is no server, and the product has very few security issues as it relates to communication. This is because there is a

security key that the client cannot access. If this product were deployed in the real world though, it would be susceptible to DDOS attacks though. This is because there is no mechanism in place that would stop a large volume of requests being sent through the program.

4. System Features

4.1 Calculation from Uploaded Picture

4.1.1 Description and Priority

This feature is the ability of the user to upload a previously taken photograph. Text within this photograph will then be interpreted, and if it is an equation the program can understand, the answer will be returned. This is of a high priority, because being able to calculate images from photographs is the central idea behind this program. Benefit would be at a level 8, penalty would be at a level 8, and cost and risk would both be at a level 2.

4.1.2 Stimulus/Response Sequences

The user will click a button that says “Upload Picture”. Then the system will switch to a new page with a prompt for the user to click to upload their picture. The user will then follow these instructions to upload a picture. The picture will then be sent to the backend of the program, which will send the image to an api to read the text from it into latex. This latex will then be interpreted, and an answer will be returned to the front end, which will display the answer.

4.1.3 Functional Requirements

REQ-1: pic-input: The ability of the program to receive photos as input is necessary for this feature to exist. If invalid input such as a non-photo file is entered, the system should return a message saying the entered file format is invalid.

REQ-2: image-parsing: The ability of the program to parse text from images is necessary for this feature to exist. If the image does not contain text, the system should display a message to the user that the image submitted does not contain any text characters.

REQ-3: calculation: The ability to calculate an equation is necessary for this feature to exist. If there is incorrect input, such as the text not being an equation, or being an equation too complicated for the program to solve, a message should be returned to the user that the system cannot solve the equation provided.

4.2 Calculation from an image in the Camera.

4.2.1 Description and Priority

This feature is the ability of the user to take a photograph using the program. Text within this photograph will then be interpreted, and if it is an equation the program can understand, the answer will be returned. This is of a medium priority, because being able to take an image within the program would greatly simplify the user experience so they do not have to separately take and then upload the image. Benefit would be at a level 6, penalty would be at a level 6, and cost and risk would both be at a level 2.

4.2.2 Stimulus/Response Sequences

The user will click a button that says “Take Picture”. Then the system will switch to a new page with a camera display and button. The user will then position the problem they wish to photograph in front of the camera and click the button. The picture will then be sent to the backend of the program, which will send the image to an api to read the text from it into latex. This latex will then be interpreted, and an answer will be returned to the front end, which will display the answer.

4.2.3 Functional Requirements

REQ-2: image-parsing: The ability of the program to parse text from images is necessary for this feature to exist. If the image does not contain text, the system should display a message to the user that the image submitted does not contain any text characters.

REQ-3: calculation: The ability to calculate an equation is necessary for this feature to exist. If there is incorrect input, such as the text not being an equation, or being an equation too complicated for the program to solve, a message should be returned to the user that the system cannot solve the equation provided.

REQ-4: Photographing: The ability to take a photo is necessary for this feature to exist because, if a photograph cannot be taken, then the user cannot take a photo of a problem using this program. If there is an error such that the camera feature stops working a message should be displayed to the user notifying them that an error has occurred preventing the camera feature connecting to the camera on their device.

4.3 Calculation from entered text.

4.3.1 Description and Priority

This feature is the ability of the user to type in an equation the program would then calculate and return the answer. This is of a high priority, because being able to calculate an equation from the text of that equation is the most basic task this calculator needs to be capable of completing. Benefit would be at a level 9, penalty would be at a level 9, and cost and risk would both be at a level 2.

4.3.2 Stimulus/Response Sequences

The user will click a button that says “Type Equation”. Then the system will switch to a new page with a textbox. The user will then type the problem they wish to have calculated. The text will then be sent to the backend where it will be converted to latex. This latex will then be interpreted, and an answer will be returned to the front end, which will display the answer.

4.3.3 Functional Requirements

REQ-3: calculation: The ability to calculate an equation is necessary for this feature to exist. If there is incorrect input, such as the text not being an equation, or being an equation too complicated for the program to solve, a message should be returned to the user that the system cannot solve the equation provided.

REQ-5: Text-entry: The ability to type an equation is necessary for this feature to exist. If the user’s entry is invalid a message should be returned to the user that the equation entered is not a valid equation that this program is capable of solving.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The interpretation and calculation of data input by the user must take no more than a few seconds because otherwise, the user will begin to wonder if the system is working. Additionally a loading icon may be put in place for the short time when these processes are taking place.

5.2 Safety Requirements

Because of the simplicity of this program, and the fact that it does not store files, it does not have any safety risks, and therefore no safety requirements.

5.3 Security Requirements

No user data is collected and therefore there are no privacy issues. The issue of anything potentially being in the background of the image submitted is negated by the fact that only a record of the equation searched is saved, the photograph is not saved.

5.4 Software Quality Attributes

The program must be able to run fully on all internet browsers, on both desktop and mobile (Primarily Google Chrome, Microsoft Edge, and Safari). The site must be formatted and easy to read on both desktop and mobile.

5.5 Business Rules

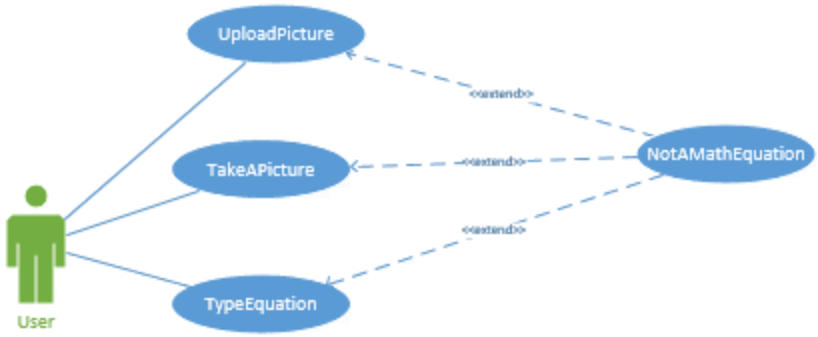
Any individual can use the application for the purpose of solving mathematical problems. The purpose of this program is for users to be able to enter mathematical equations and have those equations calculated. Primarily this program focuses on users being able to calculate mathematical equations of which they take photographs. There should be no need for anyone to interact with the program except for the user.

6. Other Requirements

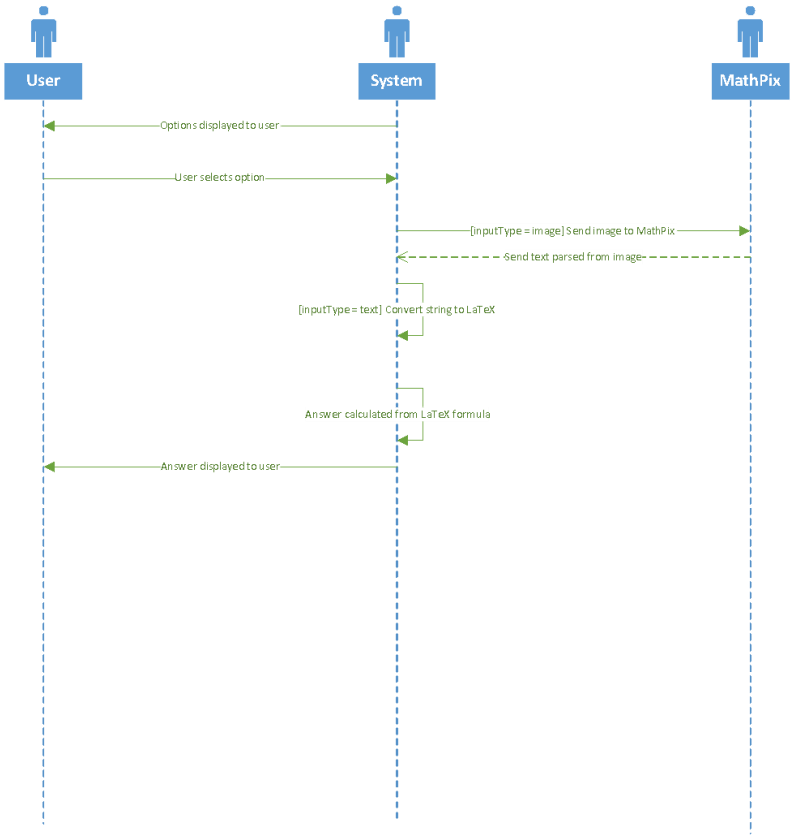
Appendix A: Glossary

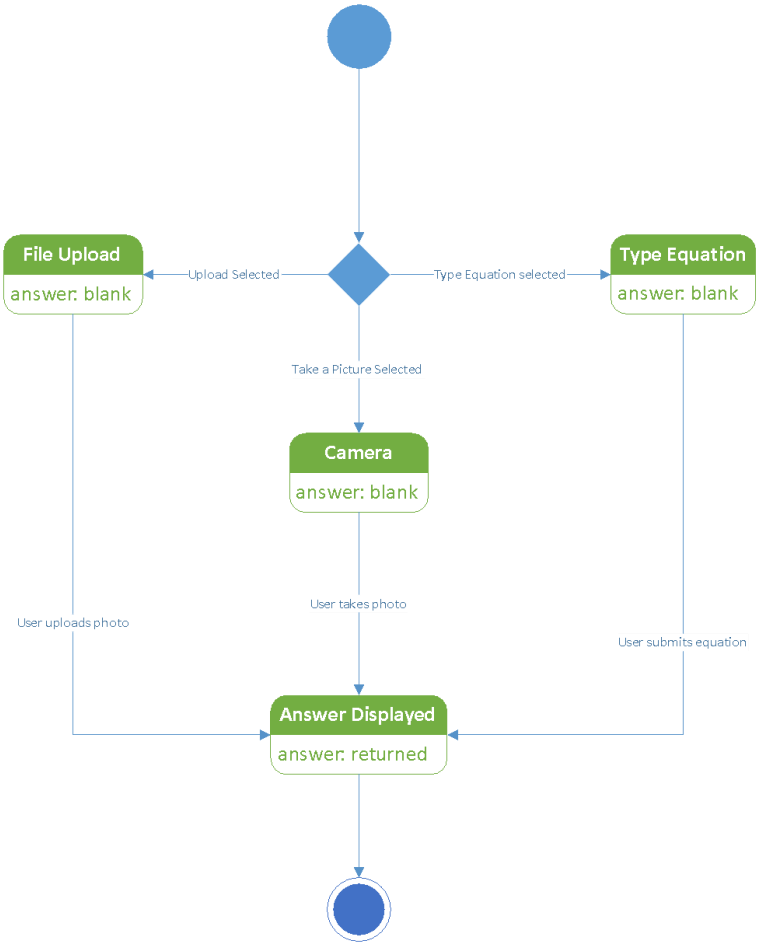
1. LaTeX: LaTeX is a plain text format that displays mathematical formulas in an easily computable manner.
2. API: APIs or Application Programming Interfaces are software interfaces that allow programs to make use of extant programs that solve complex problems. APIs are used in this product to parse the text from pictures.
3. React: React is a library for the JavaScript coding language, that makes the creation of the front end of applications easier.
4. Node.js: Node.js is a runtime environment for JavaScript, to make the development of the backend of programs simpler.
5. MathPix: MathPix is an API that parses text from images, into the LaTeX format, along with providing some other less important information to the program making use of MathPix.
6. MathQuill: MathQuill is a library for JavaScript that converts a string into LaTeX text.
7. React-dropzone: React-dropzone is a pre-made off the shelf component that allows a program to allow file upload more easily within the react framework.
8. React-webcam: React-webcam is a pre-made off the shelf component that allows for a program to make use of a camera more easily within the react framework.

Appendix B: Analysis Models


UploadPicture
<<extend>>
NotAMathEquation
TakeAPicture
<<extend>>
<<extend>>
TypeEquation
User


SolveMathEquation
MATHSCANNER SYS
ProvideSolution


User
System
MathPix
Options displayed to user
User selects option
[inputType = image] Send image to MathPix
Send text parsed from image
[inputType = text] Convert string to LaTeX
Answer calculated from LaTeX formula
Answer displayed to user



Appendix C: To Be Determined List

N/A