Software Requirements Specification Template

**BRICK FINDER**

Software Requirements Specification

April 02, 2019

Team Members

Wendy Eloe

Sachin Setty

Priyanka Bonam

Bhavika Padidala

Durga Prasd Kallem

Subhodh Bhargav Lakhinana

Submitted in partial fulfillment

Of the requirements of

CSIS 44691-01 Graduate Directed Project 1

**Table of Contents**

1.[INTRODUCTION 1](#_Toc6907773)

**1.1.**[**Purpose 1**](#_Toc6907774)

**1.2.**[**Scope 1**](#_Toc6907775)

**1.3.**[**Definitions, Acronyms, and Abbreviations 1**](#_Toc6907776)

**1.4.**[**References .**](#_Toc6907777)**2**

**1.5.**[**Overview 2**](#_Toc6907778)

2.[GENERAL DESCRIPTION 3](#_Toc6907779)

**2.1.**[**Product Perspective 3**](#_Toc6907780)

**2.2.**[**Product Functions 3**](#_Toc6907781)

**2.3.**[**User Characteristics 3**](#_Toc6907782)

**2.4.**[**General Constraints 3**](#_Toc6907783)

**2.5.**[**Assumptions and Dependencies 3**](#_Toc6907784)

3.[SPECIFIC REQUIREMENTS 4](#_Toc6907785)

**3.1.**[**External Interface Requirements 4**](#_Toc6907786)

**3.1.1.**[**User Interfaces 4**](#_Toc6907787)

**3.1.2.**[**Hardware Interfaces 4**](#_Toc6907788)

**3.1.3.**[**Software Interfaces 4**](#_Toc6907789)

**3.1.4.**[**Communications Interface 4**](#_Toc6907790)

**3.2.**[**Functional Requirements 5**](#_Toc6907791)

**3.3.**[**Non-Functional Requirements 5**](#_Toc6907792)

**3.3.1.**[**Performance 5**](#_Toc6907793)

**3.3.2.**[**Reliability 5**](#_Toc6907794)

**3.3.3.**[**Security 5**](#_Toc6907795)

**3.3.4.**[**Portability 5**](#_Toc6907796)

**3.4.**[**Design Constraints 6**](#_Toc6907797)

**3.5.**[**Database Requirements 6**](#_Toc6907798)

**3.6.**[**Prototypes (for complete project) 6**](#_Toc6907799)

4.[DESIGN 8](#_Toc6907800)

**4.1.**[**GUI: 9**](#_Toc6907801)

5.[Technical Manual 11](#_Toc6907802)

**5.1.**[**Search page 11**](#_Toc6907803)

**5.2.**[**Map Page 11**](#_Toc6907804)

**5.3.**[**Print Functionality 11**](#_Toc6907805)

**5.4.**[**Database Functionality 11**](#_Toc6907806)

**5.5.**[**Security Issues 11**](#_Toc6907807)

6.[END USER MANUAL 12](#_Toc6907808)

**6.1.**[**Search page 12**](#_Toc6907809)

**6.2.**[**Map page 1**](#_Toc6907810)**3**

**6.3.**[**Print Functionality 1**](#_Toc6907811)**4**

# **1. INTRODUCTION**

## **1.1. Purpose**

The purpose of this project is to help a user find a specific brick located in the Walk of Champions outside the Hughes Fieldhouse.

## **1.2. Scope**

This GDP team will develop and implement a working prototype for the Brick Finder system over two semesters in GDP1 in spring 2019 and GDP2 in summer 2019. This shall include:

* Reading data from an Excel file or external database,
* Implementation of search functionality within the aforementioned data,
* The creation of a map and zoning system of the brick and paver locations on the Walk of Champions.

## **1.3. Definitions, Acronyms, and Abbreviations**

**Bootstrap:** It is a library we used to make the page responsive.

**Bricks:** These are physical bricks located on the Walk of Champions. They are 4” by 8” and can contain up to three lines of personalized text.

**Data Folder**: It has the Excel sheet from where we are reading the data.

**GDP:** Graduate Directed Project

**Hughes Fieldhouse:** Located on the west side of the Northwest Missouri State University campus.

**Lib Folder:** It has all the libraries that are generated automatically when we push the code in to the GitHub repo.

**Pavers:** These are physical bricks located on the Walk of Champions. They are 8” by 8” and can contain up to six lines of personalized text.

**Visual Studio Code:** Visual Studio Code is a source-code proofreader created by Microsoft for Windows, Linux and macOS. It incorporates support for investigating, installed Git control, sentence structure featuring, smart code consummation, bits, and code refactoring.

**Walk of Champions:**  The sidewalk located to the east of the Hughes Fieldhouse where the bricks and pavers are located.

**Zone Number:**  The Walk of Champions is divided into ten zones for mapping purpose, 1 through 10 North and 1 through 10 South.

## **1.4. References**

*Visual Studio code developer documentation*. (2019). Retrieved from:

<https://code.visualstudio.com/docs>

## **1.5. Overview**

The Brick Finder system will help the user search for the location of a brick or a paver by entering the name of the donor(s) or the inscription on the brick or paver in the text field. Once the user clicks on the search button, they will see a table that contains the matching results. The user can find the specific name or inscription they are looking for and can click on that particular row. By doing so, they will see a map with the zone number where that specific brick/paver can be found and an image of a brick/paver with the inscription text overlaid on it. If the user wants to print the page, they can do so by clicking on the print button that will appear on the page.

# 

# **2. GENERAL DESCRIPTION**

## **2.1. Product Perspective**

The product perspective is to make user’s search for their brick/paver in the Hughes Field House. This solution is very helpful for donor and the administrator. They can easily identify the location and take a print of the map.

## **2.2. Product Functions**

1. Gather data from a multi-sheet Excel file.
2. Search through the data for a specific word or phrase as specified by the user.
3. Selection of a specific element (brick or paver) from a search result.
4. Display the correct zone number, map, and inscription for a specific result.
5. Print the map, zone number, and inscription for a specified result.

## **2.3. User Characteristics**

We expect the majority of the users of Brick Finder to fall into one of two categories: someone who has donated a brick or paver to the alumni organization or someone who has had a brick or paver engraved in honor of remembrance, a friend or family member, or even an affiliated organization.

## **2.4. General Constraints**

There were many constraints for this project. While not an exhaustive list, these pertinent constraints influenced the design of the project.

* No PHP
* Ability to explain how everything working in order to hand off this implementation to a different group.
* Encouraged to avoid any server-side code or database.
* If possible, use the Excel file that the Alumni group already has with brick, paver, and donor data as a database.
* Only two semesters to work on this project.

## **2.5. Assumptions and Dependencies**

* The Excel data file column names will always be the same or will be changed accordingly in the code.
* The Brick Finder System will not need offline service.
* The Excel data file used does not contain private information that needs to be secure.
* Future teams will manually update the Excel data file in the Excel program when there is new information.
* There will not be any write functionality between the Brick Finder System and the Excel data file.

# **3. SPECIFIC REQUIREMENTS**

## **3.1. External Interface Requirements**

### **3.1.1 User Interfaces**

The initial required user interfaces in this project are Search Screen and Map Screen. The user interface must be responsive and work on all web browsers including those on mobile devices.

**Search Screen:**

* Appropriate NWMSU branding and integration with the Institutional Graphic Standards
* Appropriate greetings and instructions when required
* A text box where a user can input search terms
* A button to begin the search
* A table that populates with the accurate search results
* A button to reset the search
* Navigation buttons as required

**Map Screen:**

* Appropriate NWMSU branding and integration with the Institutional Graphic Standards
* Appropriate greetings and instructions when required
* A map image displayed with the appropriate zone location highlighted
* Display of the appropriate inscription and zone number
* A button to print the inscription and map
* A button to reset the search
* Navigation buttons as required

### **3.1.2. Hardware Interfaces**

This Brick Finder project does not interface with any specialized external hardware.

### **3.1.3. Software Interfaces**

This Brick Finder project does not interface with any specialized external software.

### **3.1.4. Communications Interface**

The Excel data file is retrieved using HTTP.

**Outlook**: It is Microsoft email administration. We use outlook to send and get emails through which we communicated.

**3.2. Functional Requirements**

1. The system shall load the data from the Excel data file into a format useable by the Brick Finder system.
2. The system shall accept a user-entered search term to query through the processed data.
3. The system shall populate a selectable table with the aforementioned search results containing donor name, inscription, and zone number.
4. The system shall display the correct map that corresponds to the selected element.
5. The system shall display the correctly formatted inscription to the screen.
6. The system shall print a page containing the correct map and formatted inscription.

## 

## **3.3. Non-Functional Requirements**

### **3.3.1. Performance**

We have no specific measurable criteria as defined by the client for performance for this project. We strive to make the system as fast as possible in order to improve user experience.

### **3.3.2. Reliability**

We have no specific measurable criteria as defined by the client for the reliability of this project. We will retrieve and display the information exactly as written in the Excel data file. For our page, it is the measure of time the page can run the way toward looking through the data dependent on the information given and showing the outcomes.

### **3.3.3. Security**

For the purposes of this project, we have assumed that the information contained within the Excel data file is public knowledge and the content of the file does not need to be secret or secured. The file itself must not have write access to it and must be hosted on a secure server such as a NWMSU server, a GitHub repository, or similar. The file will need to be occasionally updated by someone with clearance to access where the data file is located.

**3.3.4. Portability**

The client specifically requested for the Brick Finder system to be responsive. We have designed this system to be accessible by the most common web browsers regardless of the device used (computer, laptop, mobile phone, etc.).

## **3.4. Design Constraints**

There were many constraints for this project. While not an exhaustive list, these pertinent constraints influenced the design of the project.

* No PHP
* Ability to explain how everything working in order to hand off this implementation to a different group.
* Encouraged to avoid any server-side code or database.
* If possible, use the Excel file that the Alumni group already has with brick, paver, and donor data as a database.
* Only two semesters to work on this project.

## **3.5. Logical Database Requirements**

We did not use a database server to store the data for this project. We did use an Excel data file to store all of the data needed for the project and are using it like we would use a database server.

The Excel file contains two different sheets, one for bricks and one for pavers. Both contain the following row headings:

* BrickNumber - This is a number given to each brick and paver. This is internally used by the Alumni organization and not the project group. There are overlapping numbers between bricks and pavers. These numbers should not solely be used for identification purposes.
* DonorName - The name(s) of the monetary donor(s) of this brick or paver.
* Personalization - The specific inscription listed on a brick or paver.
* Zone - This is the zone location of where this brick or paver is physically located on the Walk of Champions.

Each Excel sheet also has columns that list what is located on every line of the brick or paver. The brick sheet has a column for Line1, Line2, and Line3, but not every brick uses every available line. The paver sheets has a column for Line1, Line2, Line3, Line4, Line5, and Line 6 but not every brick uses every available line.

## **3.6. Prototypes (for complete project)**

We created many prototypes in many different categories of prototypes for the project. Some of them are listed below.

**Categories of Prototypes:**

**Proof of Concept Prototype:** We had 5 different proof of concept prototype assignments during the first semester of GDP.

1. Reading from a multi-tabbed Excel file from JavaScript; Open an HTML file with JavaScript. The headings from two tabs in an Excel file are displayed in the HTML.
2. Develop search algorithm for finding rows in the Excel file; Type any text (donor name or inscription). HTML page is returned containing the number of rows containing the text.
3. Type text in a search box; An HTML page is returned showing the rows in Excel that contain that text.
4. A user types in text to find. An HTML page is returned showing the rows in Excel containing that text. Each row contains a hyperlink. When clicking on that hyperlink, a new page is displayed containing the brick's inscription the brick's location and a map.
5. In the map/location page, a user can click a button to print a PDF.

**A User Experience Prototype:** It represents enough of the appearance and function of the product that it can be used for user research.

**Working Prototype and Functional Prototype:** As we finished each proof of concept, we added it to the combined working and functional prototype.

**Visual Prototype:** represents the size and appearance, but not the functionality, of the intended design. A Form Study Prototype is a preliminary type of visual prototype in which the geometric features of a design are emphasized, with less concern for color, texture, or other aspects of the final appearance.

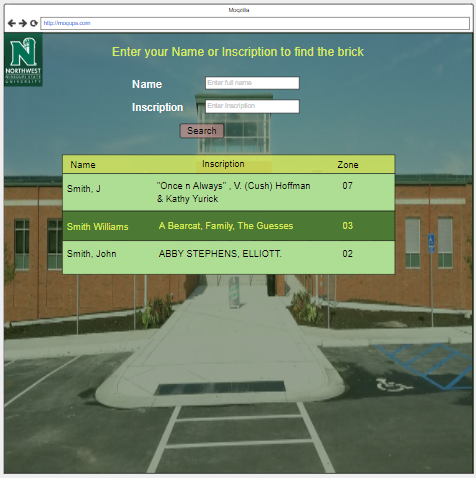
# **4. DESIGN**

## **4.1. GUI:**

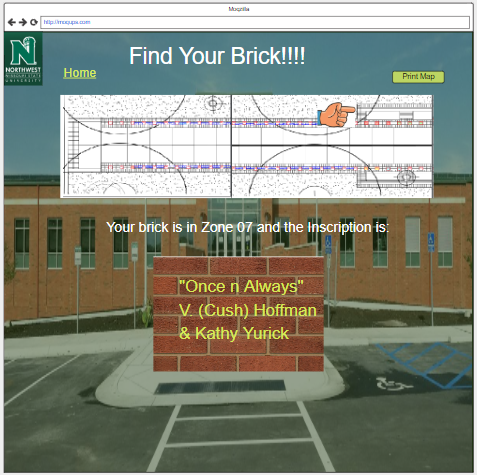
Mockups reflect the design choices for color schemes, layouts, typography, iconography, the visuals of navigation, and the overall atmosphere of the product.

**Web Site:**

Search Screen:

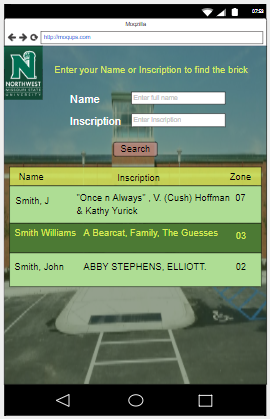
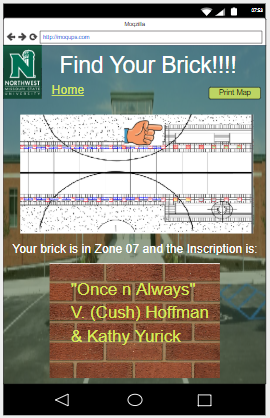


Map Screen:



**Example of responsiveness in Mobile Application:**

Search screen: Map screen:

# **5. Technical Manual**

## **5.1. Search page**

This page has Donor name or inscription text field. When the donor enters name or inscription it displays the search results in a table.

* We have UI validations for donor name and will display proper feedback about the count of the search results.

## **5.2. Map Page**

Once the user clicks on any of the search result they should be able to see the map image with the corresponding zone numbers, and a brick/paver with inscription on it.

* We used **showImage()** function to display the map on the search page by hiding the table.

## **5.3. Print Functionality**

If the user wants to take a print of the map image, they can print it by clicking the print button or they can even save the page as a PDF.

We used **window.print()** function that prints the whole page with the map and a brick/paver with inscription on it.

## **5.4. Database Functionality**

We did not use any database for this project. Everything is made client side. The client can manage the excel sheet anytime without any issues.

## **5.5. Security Issues**

There are no security issues with this project as we are not using any personal information of the donors. Client can use any public repo to store the excel sheet.

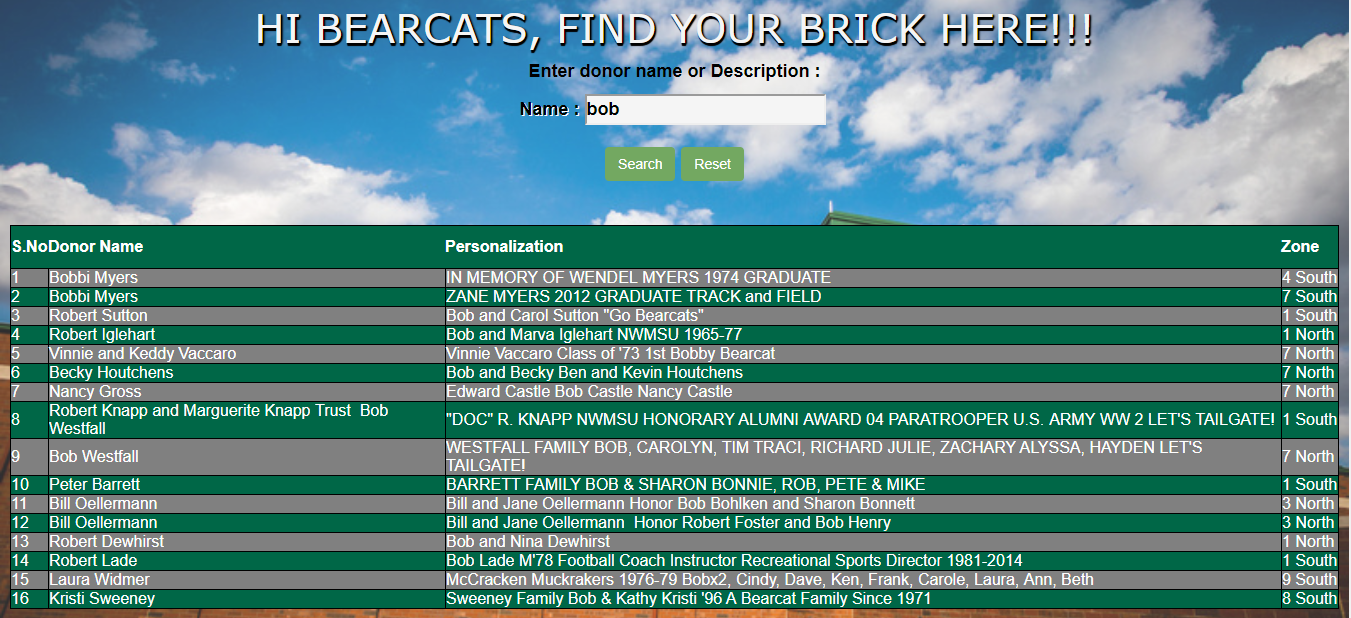
# **6. END USER MANUAL**

**6.1. Search page**

The search page allows the donors to search for their brick/paver location in the Hughes Field house by entering either the name or inscription in the text label. Once the donor clicks on the search button a list of matching results will be displayed with donor name, personalization and zone number.



Displaying the search results:



**6.2. Map page**

Once after the donor finds their name in the table they can click on that resulted row. Clicking on a particular row will display the Map pointing exactly to that zone number and a brick/paver with inscription on it.



**6.3. Print Functionality**

If the donor wishes to take a print of this map they can take it by clicking on the print button.

