# Software Requirements Specification

for

# ProjectWixs

Version 1.0

**Prepared By:**

Group 02

Drayton Williams (5925342)
Jasdeep Grewal (5757828)
Cameron Hammel (5808746)
Ian LeMasters (5877667)
Curtis Honsberger (6630362)
Liam Howes (5880331)
Kieran Colaco (6054654)
Nathan Hellinga (6002620)

**4F00 Project Wixs Analysis & Design**

**Friday, February 7th, 2020**

# Table of Contents

# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| N/A | N/A | N/A | N/A |
| N/A | N/A | N/A | N/A |

# Section 1 - Introduction

## 1.1 Purpose

ProjectWixs is a Content Management System allowing for a seamless creation of single-page web pages featuring an easy-to-use drag and drop component interface. ProjectWixs aims to provide an intuitive experience for a wide range of users; from complete beginners unfamiliar with any form of web design or HTML, to veteran designers with multiple years of experience. The system will be able to have up to ten user accounts that can create and edit web pages with a range of content typical of any modern day website (such as text, images, and video). Users will be able to create, edit, and delete up to four templates and publish one at any time for live viewing.

## 1.2 Document Conventions

All document in-text citations are foot-noted in an MLA format.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for use by developers, project managers, testers, and users with sufficient knowledge in a technical background, as all may benefit from different snippets of information provided. Sections 2 and 3 (Overall Description & System Features) provide a detailed overview of system design and features that may benefit users and testers looking to obtain a more thorough understanding of the application than that provided on the ProjectWixs Help/FAQ web page. Sections 4 and 5 (External Interface Requirements & Other Nonfunctional Requirements) provide an in-depth look at internal communications present within the system architecture and the environment necessary for sufficient operation of ProjectWixs. Developers, testers, and project managers will benefit from the aforementioned section information. Diagrams and models contained in Section 6 provide an abstracted description of different aspects of the system for those who seek further information.

## 1.4 Project Scope

ProjectWixs is intended to be used by those with access to the internet who wish to create web pages without the prerequisite of understanding how to code in common web technologies such as HTML, CSS, or JavaScript. The system caters to users ranging from little-to-no experience with web development, all the way to seasoned experts who are looking for a simple UI for quickly establishing a personal web presence. System access will be restricted through a basic system security, which will be implemented through the use of a user login system. Users will be able to upload photo or video content. Overall, this system is meant to make the task of creating and publishing a web page as efficient of an experience as possible, through the use of a drag-and-drop template design system.

## 1.5 References

### 1.5.1 Initial System premise

Initial system premise (and contract specification) based on project statement as provided by the COSC 4F00 course professor Dave Bockus.
Found Here:
https://github.com/DrayWilliams1/ProjectWixs/blob/master/Project%20Statement.pdf

### 1.5.2 Initial CMS architecture

Initial CMS architecture page concepts, and feature set inspired by other popular competitors such as WordPress, Wix and Bolt CMS.
Found Here:
https://wordpress.com/ (WordPress website)
https://www.wix.com/ (Wix website)
https://bolt.cm/ (Bolt CMS website)

### 1.5.3 Password strength

Criteria for password creation based on Lafayette College's standards for a strong password.
Found Here:
https://its.lafayette.edu/policies/strongpasswords/

### 1.5.4 Minimum software versioning

Information on minimum software versioning referenced in section 4.3 is obtained from a PHP info file hosted on a shared sandcastle group account.
Found Here:
http://cosc.brocku.ca/~dw15we/4F00Test/phpinfo.php

### 1.5.5 Branding Standards

Front-end system design will adhere to a project-specific style guide regarding colour schemes, spacing, and typography. Obtained from style guide in section 6.1

### 1.5.6 Coding Standards

System will adhere to programming standards agreed upon by the W3C.
Found Here:
https://www.w3.org/standards/

### 1.5.7 Accessibility Standards

System will adhere to accessibility standards as agreed upon by the W3C.
Found Here:
https://www.w3.org/standards/webdesign/accessibility

# Section 2 - Overall Description

## 2.1 Product Perspective

To help understand why Content Management Systems (CMS) like ProjectWixs are beneficial to organizations, it is important to understand what CMS' are and what they do. In a study discussing CMS' and the organization's that adapt them, it was noted that,

> Some form of content management (CM) process or system is becoming essential for all organisations with a significant Web presence as the amount of digital content continues to proliferate... Software product companies have moved to address this need and call their offerings content management systems. Such systems help a business to set-up and organise their Web site(s), so that the Web sites can grow and change rapidly while maintaining high quality[1]

CMS' such as ProjectWixs offer an easy way for companies to develop their online presence in an increasingly digital world. The exact implementation is flexible - whether an organization needs a public site to help build their web presence, or a more interorganizational site to help share data between employees - CMS' provides the tools to make these a reality. ProjectWixs delivers this, while also allowing for the creation of clean, efficient websites with no prior knowledge of HTML or other web design experience.

ProjectWixs is a new CMS, conceived to serve as a competitor to existing web development apps such as Wix and Wordpress. Being a new product, ProjectWixs does not aim to succeed or carry over content from any existing CMS, but serve as a new, better replacement to any organization's current system. Being web-based, the only requirement for an organization to set it up would be a server with capabilities similar to Sandcastle, and Internet-connected devices for users to access the site from (See Figure B-1: Basic System Architecture).

## 2.2 Product Features

As a Content Management System, ProjectWixs will provide the adapting organization with all the tools necessary to begin creating sites and content to serve organizational goals. This begins with providing secure, controlled access to the site editing capabilities, which ProjectWixs provides through standard account management. When first setup, an initial administrator account will be available that can approve new user requests, manage the system's user base, and choose other accounts with which to share administrative privileges. Other users can then apply to register by going through the main Project Wixs home page. The administrator(s) can then choose to approve these requests, and users can then begin editing sites.

[1] "Understanding Web content management systems: evolution ...." 29 Apr. 2018, https://www.researchgate.net/publication/220672404_Understanding_Web_content_management_systems_evolution_lifecycle_and_market. Accessed 4 Feb. 2020.

All non-administrative accounts are regular user accounts, with each account being able to edit and create four user templates as well as host a private collection of web media content (images and videos) that can be shared on the site. To customize these sites, Project Wixs features a drag-and-drop editor that will allow users to create web templates with little to no knowledge of HTML. This will include standard HTML components such as headers, text, links, etc. as well as the aforementioned media content that can all be organized and arranged in a visual interface. For more experienced users, an option to switch to a plaintext HTML editor will be included as well.

Finally, ProjectWixs will feature a Help section that will help guide new users on how to use the ProjectWixs editor. When a new user first logs in, there will be a recommendation for the user to visit the help page. Access to the Help section will also be available through a link on the footer of all pages of the ProjectWixs site. This Help section will contain text instructions, helpful links, and embedded video tutorials to help the user familiarize themselves with ProjectWixs' capabilities.

## 2.3 User Classes and Characteristics

There are two specific ways of classifying user's we wish to address. The first is by the user's skill level, which is an important element to consider for what is the CMS' most central feature; its website editor. The other is a more general look across the entire site's userbase, comparing and contrasting their access and privileges.

### 2.3.1 Classification by Skill Level

The Project Wixs site editor has options for users of all skill levels. However, certain features of ProjectWixs may or may not be relevant to the user's skill's level, which we have classified below. Overall, it *will* favour Beginner and Intermediate users with limited to no HTML experience (which is common for those outside the fields of Computer Science). The drag-and-drop functionality lends itself better to these low-level users, rather than high-level users who want greater freedom of customization.

Beginner

Beginner users are those with little to no experience with HTML or web design. Overall they are likely not very technologically literate in general, and may be intimidated by any form of plain text forms of code. Project Wixs seeks to appeal to them by allowing them not to have to touch HTML code at all, and to work exclusively within the visual drag-and-drop editor. In addition, the help pages are easily accessible and provide detailed, beginner-friendly tutorials in both video and text format.

Intermediate

Intermediate users would be those with under 5 years of experience with HTML and/or web design. They mave feel confident using HTML, but could appreciate the convenience that the drag-and-drop editor provides and use a mix of both.

Expert

Expert users have over 5 years of experience with HTML/web design, and may not want to even bother with learning the drag-and-drop editor. They will likely be put off with the extra features or restraints put upon the editor, and would want full customization of the webpage. This should only require one click to move to editing plaintext HTML. This should be an easy enough switch to do, and once they are in the plaintext editor they will not need to bother with any of the extra editor features.

### 2.3.2 Classification by Privileges

Visitor

Any user to the site with no account registration. They can apply to become a user from the ProjectWixs homepage.Otherwise they are able to view any site they know the address to. These users are more interested in the end-user site’s than they are with the CMS, and are of low priority.

Editor

The editor class encompasses all the classes by skill level described in the previous classification. Editors are users that have access to edit their own webpage and media content library. These users have a wide range of needs, and the previous classification goes into detail on how each class is serviced.

Administrator

The admin class shares all the same functionality of editors, but with the addition of account management. This encompasses an additional page listing all the accounts and site statistics. While the page is still organized and clearly mapped, we expect the admin to be more comfortable with back-end elements and be able to deal with the large amount of information presented here.

## 2.4 Operating Environment

The equipment available within the Department of Computer Science at Brock University will be used to host the ProjectWixs system, and as such, will operate on the x86_64 Red Hat Linux operating system. PostgreSQL is a powerful, open source object-relational database system and will be used as the database system for ProjectWixs. The powerful server scripting language PHP (Hypertext Preprocessor) will be used to create user’s dynamic webpage content and work hand-in-hand with PostgreSQL by actively modifying files stored in its database. For all front-end aspects, React.js will be used. React.js excels specifically in building user interfaces for single page applications, which is precisely the purpose of this project, and is therefore an ideal choice.

## 2.5 Design/Implementation Constraints

In order for user's webpages to load efficiently, ProjectWixs will have a maximum allowable photo size of 5 MB. In most cases, this file size assumes responsibility of the designer to use a combination of reducing file size, reducing resolution in dpi, and image compression on their image before attempting to upload. ProjectWixs only assumes responsibility for failure to load MP4 and WebM video file formats, which are both compatible with the most common browsers: Chrome and Firefox. ProjectWixs will not support 360 degree-view or 4k resolution video. Maximum allowable video file size will be 25 MB. These limits on photo and video file sizes ensures that webpages run and load effectively as this is a common problem for beginner designers with little knowledge about image optimization for the web. This also ensures that users do not try and upload incredibly large file sizes in an attempt to disrupt ProjectWixs' services.

## 2.6 User Documentation

Users will be offered an array of tutorials and helpful documentation, including this Software Requirements Specification document. In addition to this document, short YouTube tutorial videos will be created and linked with topics including, but not limited to: user account creation and login, template creation, and editing user templates. Text-based tutorials and guides will also be provided. On user account creation, users will be prompted to click on a pop-up dialogue box that will link to the the Help page which contains all the help resources. This dialogue box may be closed and will not popup again, but the resources may be accessed again at any time through the Help button, which stays persistent in the footer throughout the whole site.

## 2.7 Assumption and Dependencies

ProjectWixs assumes that all users are not visually impaired or impaired in any other form that would reasonably render our content management system unusable or extremely difficult to navigate.  Our system will currently not implement any features for the use by those visually, or similarly impaired individuals such as auditory prompts. These individuals are of the extreme minority of the targeted user base and implementing such features would require considerable time and resources currently unavailable within the project time and resource constraints.

# Section 3 - System Features

## 3.1 Persistent page features

3.1.1 - Description and Priority

- Working left to right, every page consists of a primary header spanning the top of the page. This header persists throughout every webpage and contains buttons to transverse to the home page and dashboard; with an additional button to login or logout respectively (dependant if the user is currently logged in or not). This is demonstrated in (Figure B-6).

- In addition, every page also displays a footer at the bottom also demonstrated in (Figure B-6).This footer contains copyright information followed by a help button which brings the user to the help page.
- **Priority - Medium:** As the buttons included are not critical for system performance, but simply help the user traverse through the site.

3.1.2 - Stimulus/Response Sequences

- Button presses would take the user to their desired location. As the footer and header is static throughout every page, these buttons can be pressed on every page.

3.1.3 - Functional Requirements

- N/A

## 3.2 Landing Page/Product Demo

3.2.1 - Description and Priority

- Upon arrival to site, the user is met with the landing page. The landing page presents our logo front and center, below the header. Below the logo, visitors can find 3 examples of features Figure B-6. The page provides basic high-level information about key system features and provides some interaction for those wishing to visit the website without fully engaging with the application
- **Priority - Medium:** Presentation is key and this page provides a quick way of displaying the system in a clean manner without much work required on the users end. Not vitally necessary to system function, however, we see it as important for a complete overall system.

3.1.2 - Stimulus/Response Sequences

- Visiting the website URL (to be decided)

3.1.3 - Functional Requirements

- N/A

## 3.3 User Registration/Login

3.3.1 - Description and Priority

- Navigating to the login page by means of the login button on the top right of the header presents the user with a screen to login to their account (Figure B-8). This screen contains two fields for input; an email/username followed by a password field. Ideally, there would also be a button underneath to register for a new account.

- The register page (Figure B-8) looks similar to the login page but would have two additional fields. This time, there would be independant fields for username and email. In addition, there would also be an input field for the user’s password and another input field to ensure the previously written password is typed the same.
- **Priority - High**: As the majority of system features rely on a user interacting with their own individual templates, it’s essential that the system recognizes users individually and serves them the right content.

3.3.2 - Stimulus/Response Sequences

- In terms of the login page, after the login button is pressed and a user is verified, they are brought to the users unique template selection screen (Figure B-5).
- Similarly, with the register page, the register button would confirm the user input, create the account, and redirect to the template selection screen (Figure B-5).

3.3.3 - Functional Requirements

- **Communication with PostgreSQL database** - In order to verify the user information has been correctly obtained or submitted to the server for authentication

## 3.4 User Template Selection

3.4.1 - Description and Priority

- After a user is verified through the login page they are redirected to their own template selection screen (Figure B-5). This screen greets the user by name and displays their three active templates. Each template block provides a field to edit the template name followed by two buttons below to edit said template or delete said template.
- An admin button is included in the top right of the screen below the header if the current logged in user is an admin. This button brings the admin to the Admin tools page (Figure B-4).
- **Priority - High**: As the main divider between users and their templates, this screen is important as it’s the gateway to the user’s content.

3.4.2 - Stimulus/Response Sequences

- Each edit button would bring the user to the template editor page for said template (Figure B-7).
- Delete buttons would pull up a prompt which would require further input to delete said template.

3.4.3 - Functional Requirements

- **Communication with PostgreSQL database** - In order to obtain and display the correct templates for the specified user.

## 3.5 Template Creation/Edit

3.5.1 - Description and Priority

- Once a user selects a template to edit from the template selection screen (Figure B-5), they are brought to the template editor page for said template (Figure B-7). On this page, the current template is shown as a preview, along with a component library on the right which allows the user to drag and drop components into the preview window.
- **Priority - High**: Arguably the highest priority in terms of utility, this screen is important as it's the main feature for users.

3.5.2 - Stimulus/Response Sequences

- Dragging and dropping a component from the item pallet on the right to the preview on the left would place said component where the user dropped it.
- The preview will confirm the layout and media chosen upon placement within that area.
- Once the user is content with their preview they are able to publish their site by means of the publish button on the left side of the detail editor header.

3.5.3 - Functional Requirements

- **Communication with PostgreSQL database** - Calls upon a pre-existing database template and allows the user to drag and drop their own documents and files into the template. A separate instance of that edited template is then saved as a user template.

## 3.6 Help Section and video guides

3.6.1 - Description and Priority

- To help with user interaction, the help page (Figure B-4) acts as a tool for those who require help with formatting and creation of their site. Ideally, a tutorial video will be embedded onto the page providing a visual guide on features. In addition, a text guide will be available beside the video, along with additional helpful links below the video.
- **Priority - Low**: Help pages serve to assist. Although it would be nice to have, it's not a critical component for user use, it serves as an aid for those in trouble.

3.6.2 - Stimulus/Response Sequences

- The help page can be reached by pressing the help button in the footer. Which would bring them to (Figure B-4).

3.6.3 - Functional Requirements

- N/A

## 3.7 Publish User Template

3.7.1 - Description and Priority

- Publishes the user's template as a fully-functional, active webpage
- **Priority - High**: Without this feature being functional, ProjectWixs would fail on delivering its purpose to its user base.

3.7.2 - Stimulus/Response Sequences

- A Publish button should be available outside of the template editor, on the template selection screen (See Figure B-5), as well as within the editor itself (See Figure B-7).

3.7.3 - Functional Requirements

- **Communication with PostgreSQL database** - database must correctly host the final, saved version of the user's template.

## 3.8 Delete User Template

3.8.1 - Description and Priority

- From the template selection page (Figure B-5), users will be able to delete existing templates as they see fit.
- **Priority - Medium**: it lies below *Template Creation/Edit* in terms of importance, but is still necessary to keep the template selection page as clean as possible.

3.8.2 - Stimulus/Response Sequences

- Templates may be deleted by pressing a "Delete Template" button underneath the desired template.

3.8.3 - Functional Requirements

- **Communicate with PostgreSQL database** - queries for the pre-existing database template and removes it.

## 3.9 Load site statistics (Admin)

3.8.1 - Description and Priority

- A page for system administrators allowing for the viewing of current active users and any other relevant statistics.
- **Priority - Low**: Statistics will be useful for administration, but not critical to the core Wixs experience.

3.8.2 - Stimulus/Response Sequences

- Templates may be deleted by pressing a "Delete Template" button underneath the desired template.

3.8.3 - Functional Requirements

- **Communication with PostgreSQL database** - queries the database for relevant site statistics and existing user accounts.

# Section 4 - External Interface Requirements

## 4.1 User Interfaces

Although implementation for ProjectWixs has not officially commenced, the system's front-end will still follow generally defined guidelines. Concept images for basic page layout can be found in Appendix B under Concept Page Layouts and a styling guide dictating colour pallette, typography, and spacing can be found in Section 6. Majority of the navigable web pages present in the system will include shared components such as the project logo, header, and footer areas in order to maintain consistency. Based on HCI concepts, all inputs and operations will be designed to provide users with accurate feedback on what just occurred. Where possible, tooltips (when hovering over links) will be implemented to ensure ease of use. Body text will always be at an adequate contrast to the background of its respective container for a better viewing experience. Currently, there are no planned keyboard shortcuts for use within the system. The webpage will not make use of third-party ads or tracking systems at this time.

## 4.2 Hardware Interfaces

ProjectWixs' front-end interface will be viewable from any device able to use an internet browser, however, the interface design will be best optimized for large-screened devices like Laptops and Desktop computers. With the system being hosted on Brock University's Sandcastle, this server will be the main hardware component controlling server-side operation of the database and hosting of system-relevant files. Beyond the aforementioned devices, the system does not interact with other forms of hardware.

## 4.3 Software Interfaces

In order for the front-end interface to display and operate as intended, JavaScript must be enabled within the client browser. Depending on the convenience of operations, small helper components may be imported (and adequately attributed) for the lower-functioning tasks of the system. For example, to facilitate the use of the AJAX technique, Axios[2] may be used alongside React.js to enhance the process. Time permitting, Google's reCaptcha may be introduced to better the security of the login system.

**The system will operate on the following back-end technologies**:

- Red Hat Linux
- PHP v5.4.16 (minimum)
- PostgreSQL v9.2.24 (minimum)

**The system will operate on the following front-end technologies**:

[2] "axios - npm." 22 Jan. 2020, https://www.npmjs.com/package/axios. Accessed 4 Feb. 2020.

- HTML - DOM HTML support enabled
- CSS
- JavaScript
- React.js

### 4.4 Communication Interfaces

The system will make use of various standards and protocols in order to ensure accurate delivery and processing of information. Being hosted on the web, ProjectWixs will follow HTTP and TCP/IP protocols for data transfer. To facilitate the generation of dynamic web pages, PHP will be utilized through the CGI protocol. HTTP interactions will be handled with AJAX technique for asynchronous data exchange. In order to connect to the Brock Sandcastle server, the system (and the team) will make use of the SSH protocol to allow for secure client-side access to the server over an unsecured network. In order to communicate with the database, queries will be formatted to follow PostgreSQL syntax. User passwords will be MD5 hashed before transfer to the database.

## Section 5 - Other Nonfunctional Requirements

### 5.1 Performance Requirements

In order to facilitate timely page loads and data transfer, the system's front-end architecture will make use of the AJAX technique for asynchronous client and server-side communications; this is crucial for system response without page refreshes and dynamically displaying data. No process should take over a minute (60 seconds) of time to complete, whether displaying, saving, or retrieving system data. Any tutorials or user-guidelines necessary to facilitate the use and mastery of the published system will be available on the Help Page (See Section 3.6).

### 5.2 Safety Requirements

In order to keep the system contained and with a minimal impact on host servers, quantifiable limits will be put in place for various different features. The file size of content uploads will be restricted to 5MB for photos and 25MB for videos. Up to 10 accounts may be registered within the system at any time (with one being reserved for an administrative account). Each of these user accounts may have up to four (4) custom templates stored at any time. For a documented safe intended use of the system, a tutorial/walkthrough will be provided (See Section 3.6).

### 5.3 Security Requirements

In general, system front-end access will be limited (with necessary page redirects) to those who have a registered account in the database. In order to maintain user privacy and security, a number of measures are being taken. User uploaded files will be matched with a unique identifier to them and only viewable (or usable in templates) solely by themselves. Upon account creation, user passwords will be hashed and stored in the database so that those with direct database access cannot obtain the password. The system will also require that user created passwords, first meet a specified criteria to ensure password safety (See Section 1.5). Password inputs will be sanitized to check for injection attacks before account confirmation and submission into the database. Authenticated routes for site data display and user account administration will be available only to accounts with admin permissions.

## 5.4 Software Quality Attributes

### 5.4.1 Maintainability

A set team member or two will be in charge of ensuring files are up to standard, however, for the duration of this project the system will be rotationally maintained (whether it be commenting, bug fixes, or algorithm optimization) by all members of the ProjectWixs team as necessary. Where possible, all source code files will contain a comment block including project name, description of file processes, and system version number. To keep a simple architecture, the system will run on as minimal an amount of files as possible, while maintaining functionality. In addition, multiple iterations of the files controlling system design will exist within a shared GitHub repository.

### 5.4.2 Availability

Considering this is meant to act as a personal website for most, the system will be expected to display 24/7. As the system will be hosted on Brock's Server, its uptime (and overall availability) will mirror that of Sandcastle's.

### 5.4.3 Reliability

As mentioned in Section 5.2, various validity checks will be performed on user input to ensure it is of an adequate size, representation, and reasonability. Exception handling will be implemented for possible null values, invalid data or overflows.

### 5.4.4 Portability

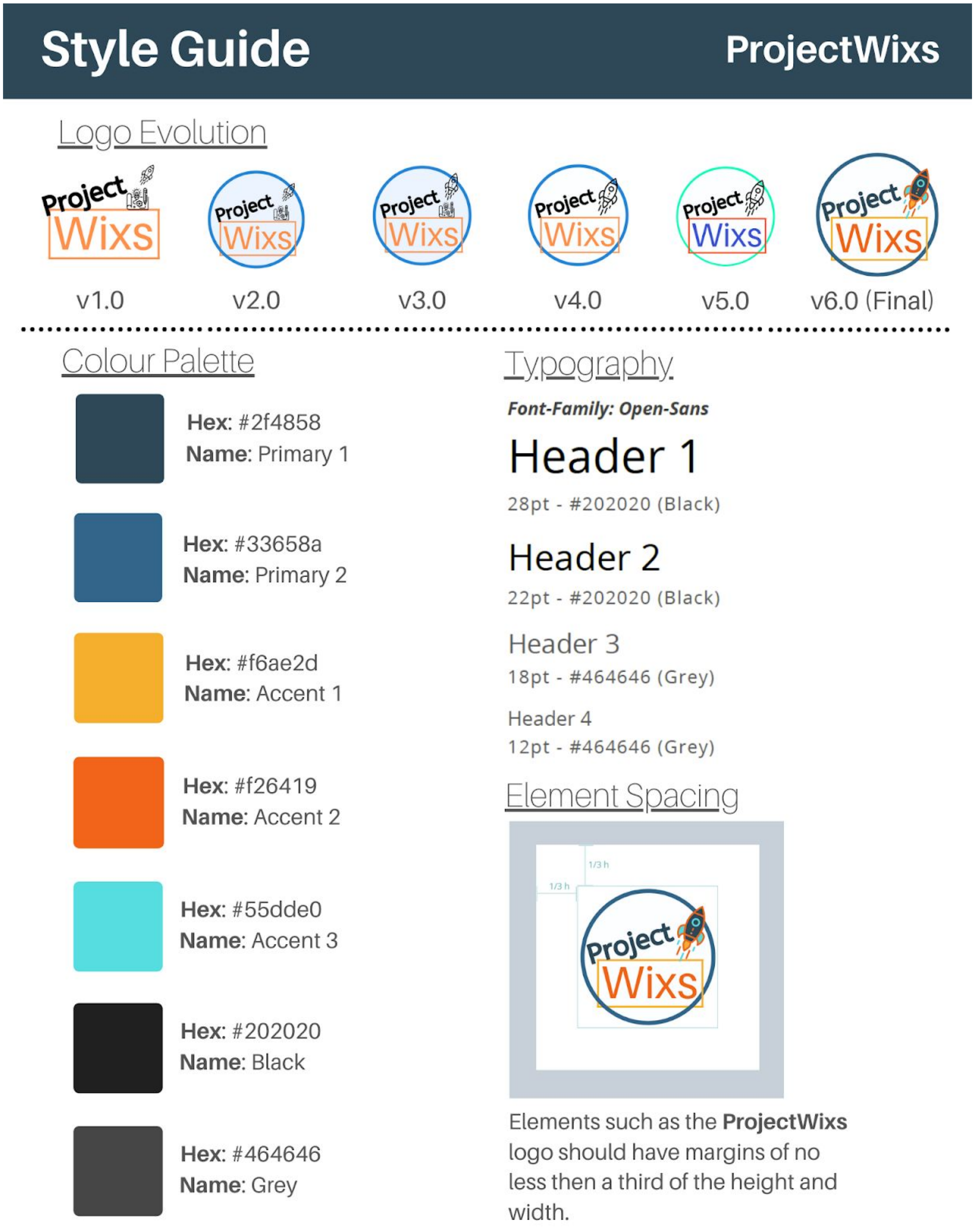
The systems software will have a relative ease when it comes to portability as it will make use of languages with a wide coverage of support across platforms, hosting environments, and compilers. 0% of the components in the system will be dependant on the host server being used (in this case, Sandcastle). All languages being used for the front-end and back-end are widely used and highly portable. ProjectWixs' main front-end language, React.js, is one of the more recently introduced languages. Based on various statistics[3], it is clear React.js is one of the most popular and well supported front-end javascript frameworks to use right now. The main back-end language, PHP, has been a staple of server-side programming for decades. It is currently maintained that PHP is used by 79% of websites with a known server-side language[4].

[3] "2019 Stats on Top JS Frameworks: React, Angular ... - Tecla.io." 10 Jun. 2019, https://www.tecla.io/blog/2019-stats-on-top-js-frameworks-react-angular-and-vue/. Accessed 6 Feb. 2020.

[4] "Usage Statistics and Market Share of PHP for ... - W3Techs." https://w3techs.com/technologies/details/pl-php. Accessed 6 Feb. 2020.

# Section 6 - Other Requirements

## 6.1 Style Guide



Style Guide

ProjectWixs

Logo Evolution

v1.0

v2.0

v3.0

v4.0

v5.0

v6.0 (Final)

Colour Palette

**Hex**: #2f4858
**Name**: Primary 1

**Hex**: #33658a
**Name**: Primary 2

**Hex**: #f6ae2d
**Name**: Accent 1

**Hex**: #f26419
**Name**: Accent 2

**Hex**: #55dde0
**Name**: Accent 3

**Hex**: #202020
**Name**: Black

**Hex**: #464646
**Name**: Grey

Typography

***Font-Family: Open-Sans***

Header 1

28pt - #202020 (Black)

Header 2

22pt - #202020 (Black)

Header 3

18pt - #464646 (Grey)

Header 4

12pt - #464646 (Grey)

Element Spacing

Elements such as the **ProjectWixs** logo should have margins of no less then a third of the height and width.

# Appendices

# Appendix A - Glossary

## Acronyms/Abbreviations

**AJAX**: Asynchronous JavaScript and XML
**CSS**: Cascading Style Sheets
**CGI**: Common Gateway Interface
**CMS**: Content Management System
**HTML**: Hypertext Markup Language
**IE**: Internet Explorer
**MB**: Megabyte (~1,000,000 bytes)
**Postgres**: PostegreSQL relational database
**SSH**: Secure Shell
**UI**: User interface
**UX**: User experience
**W3C**: World Wide Web Consortium
**x86-64**: the 64-bit version of the x86 instruction set.

## Definitions

**Chromium**: An open source web browsing software by Google.[5]
**Sanitize/Sanitizing**: The process of cleansing and scrubbing of user input to prevent it from jumping the fence and exploiting security holes.[6]
**Sandcastle**: Brock University Computer Science department's primary hosting server.[7]

[5] "Chromium - The Chromium Projects." https://www.chromium.org/Home. Accessed 29 Jan. 2020.
[6] "Prevent Web Attacks Using Input Sanitization - eSecurity Planet." 26 Oct. 2012, https://www.esecurityplanet.com/browser-security/prevent-web-attacks-using-input-sanitization.html. Accessed 28 Jan. 2020.
[7] "HOW-TO Index - Computer Science - Brock University." https://www.cosc.brocku.ca/help/howto. Accessed 6 Feb. 2020.

# Appendix B - Analysis Models

## System Architecture

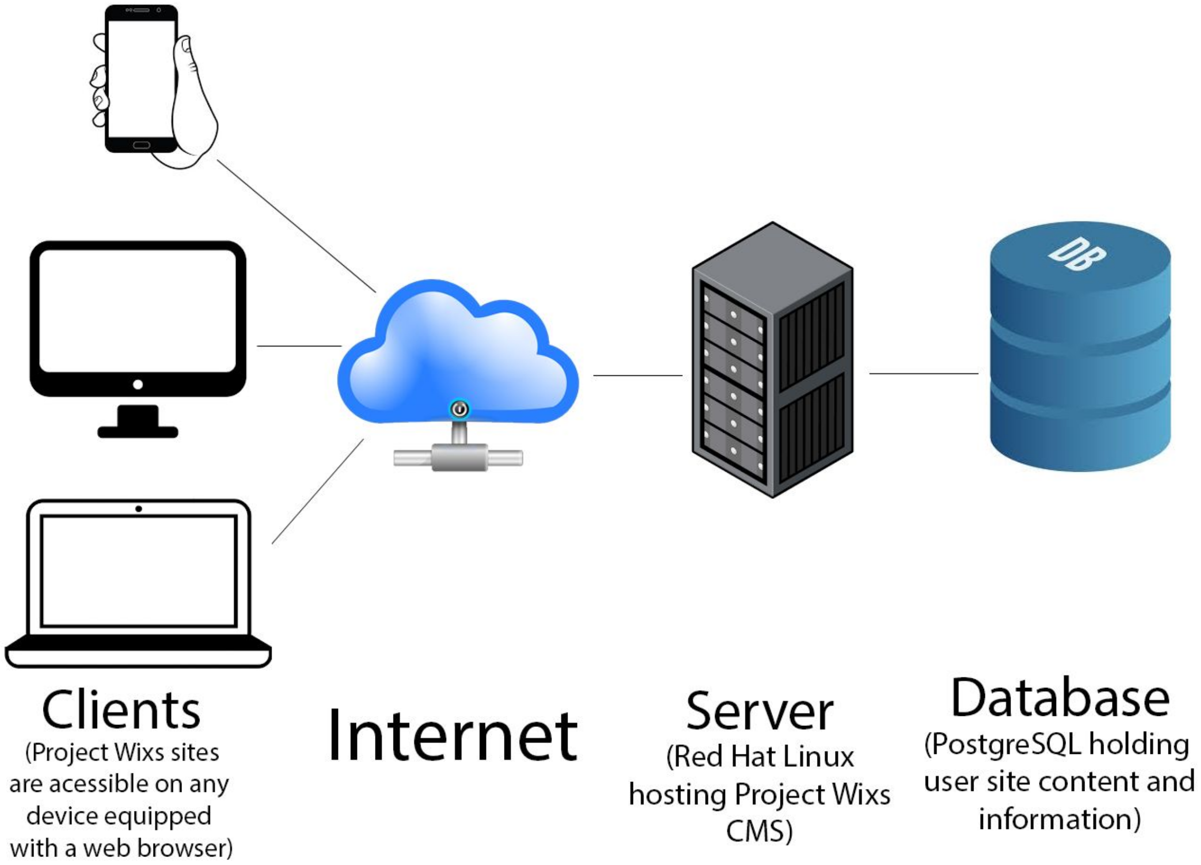
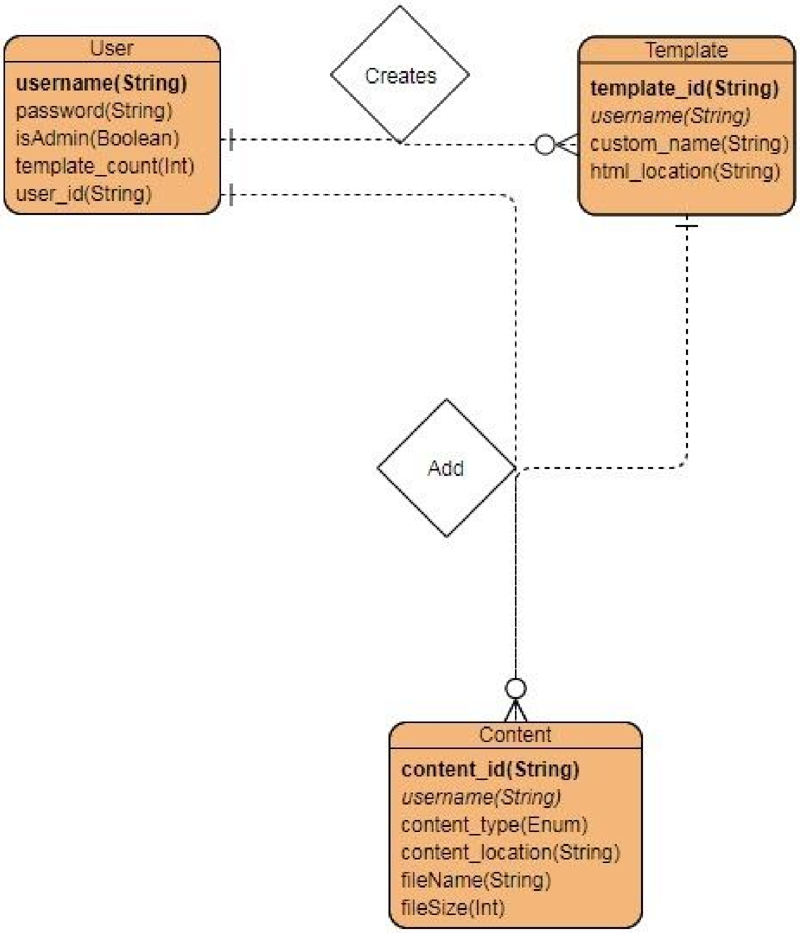

DB
Clients
(Project Wixs sites are acessible on any device equipped with a web browser)
Internet
Server
(Red Hat Linux hosting Project Wixs CMS)
Database
(PostgreSQL holding user site content and information)

**Figure B-1: Basic System Architecture**

## ER Model Diagram



- **Bolded Attributes** = Primary Key (username, template_id, content_id)
- *Italicized Attributes* = Foreign Key (username)

**Figure B-2: ProjectWixs ER Diagram**

## Use Case Diagrams

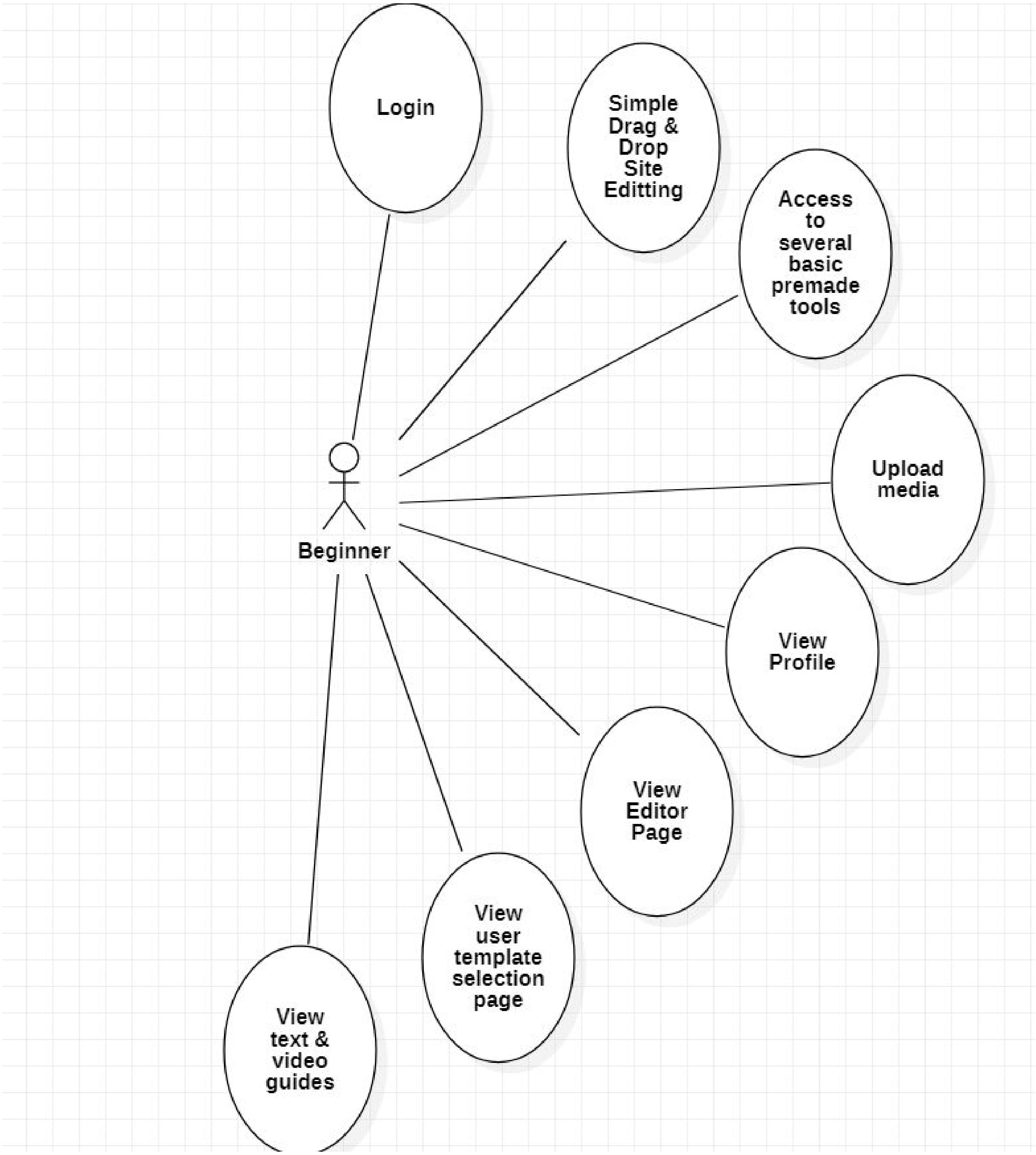

Login
Simple Drag & Drop Site Editting
Access to several basic premade tools
Upload media
View Profile
View Editor Page
View user template selection page
View text & video guides
Beginner

**Figure B-3: Use Case Diagram for a Beginner user**

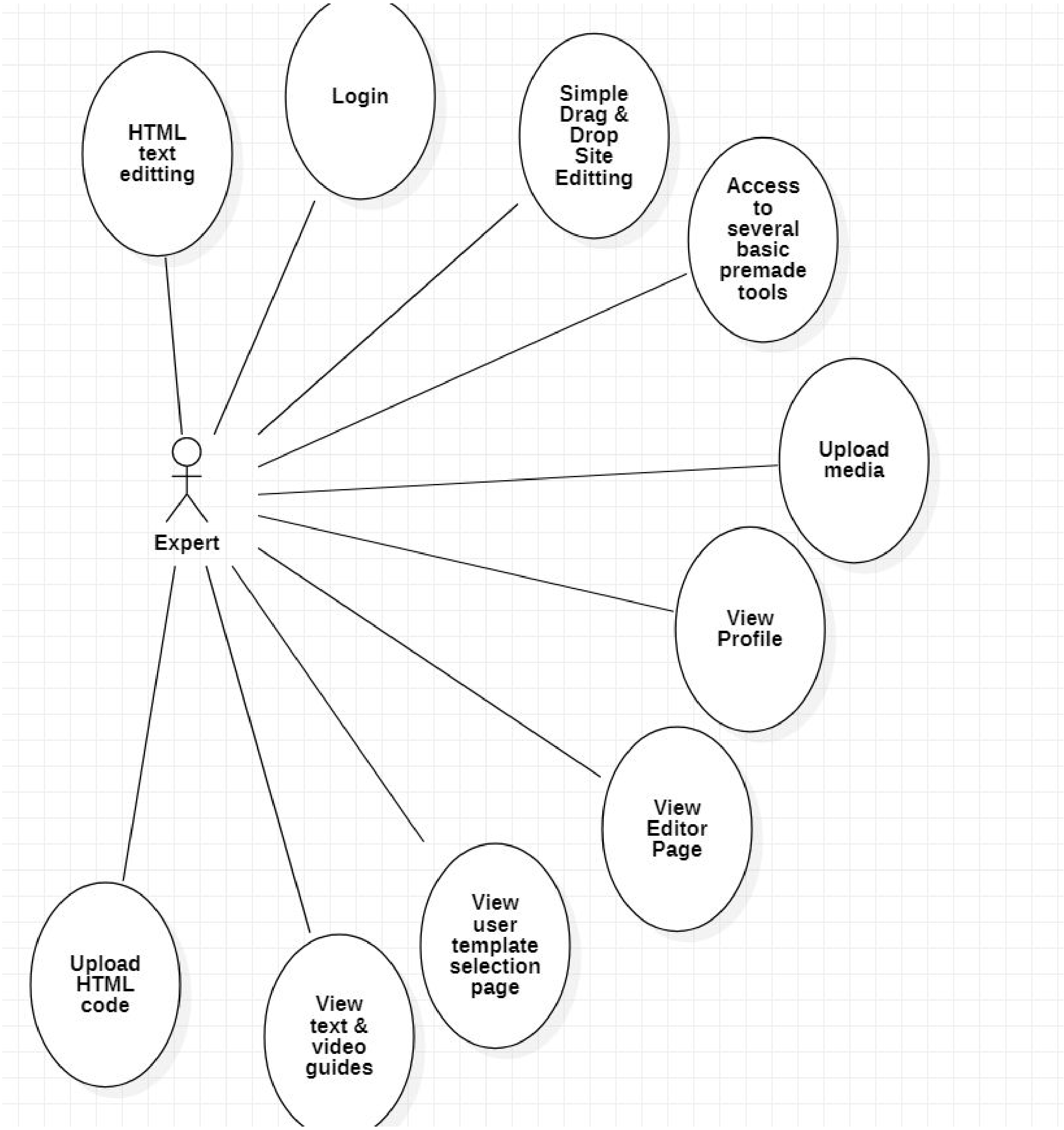


**Figure B-4: Use Case Diagram for an Expert user**

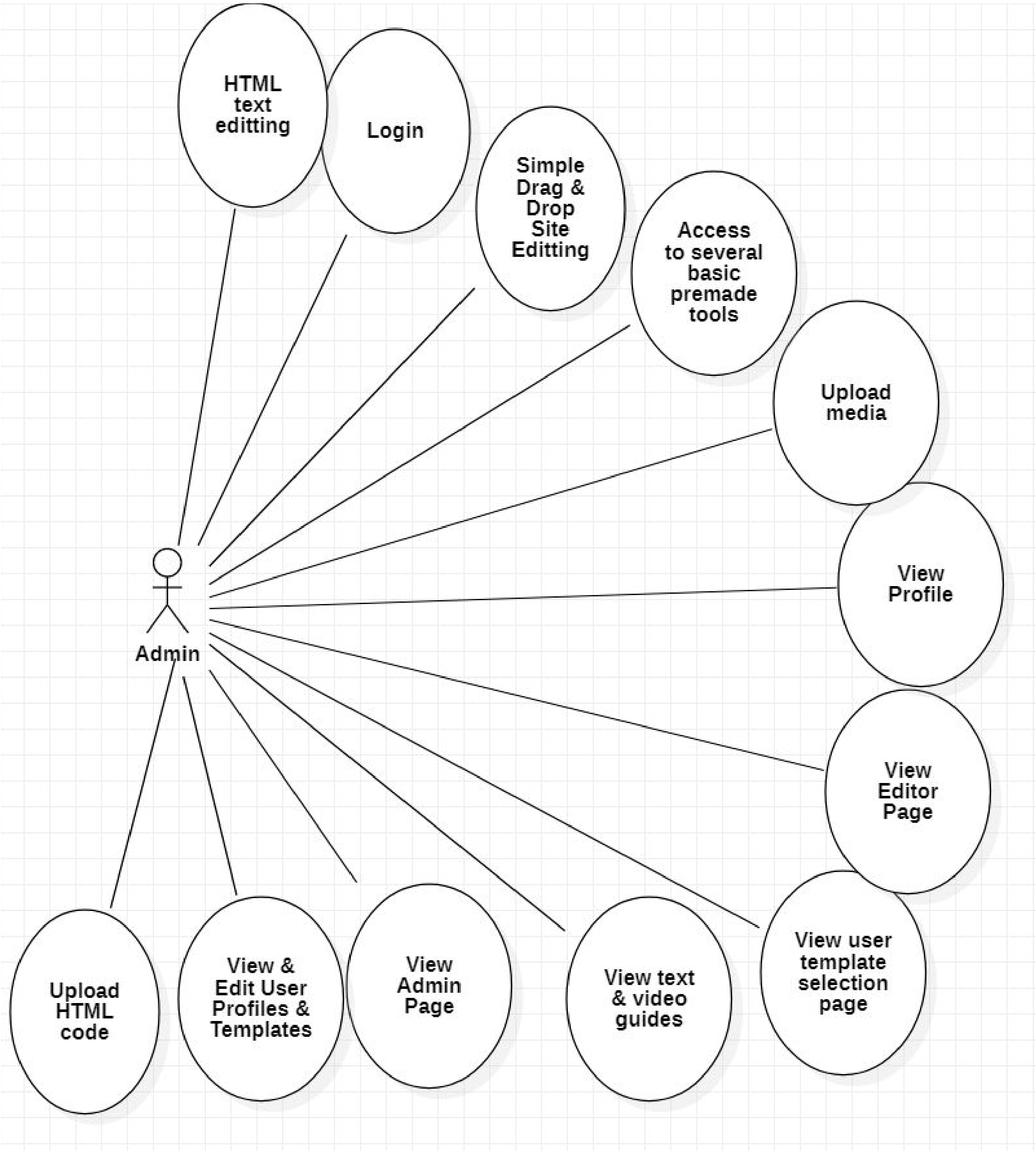

HTML text editting
Login
Simple Drag & Drop Site Editting
Access to several basic premade tools
Upload media
View Profile
View Editor Page
View user template selection page
View text & video guides
View Admin Page
View & Edit User Profiles & Templates
Upload HTML code
Admin

**Figure B-5: Use Case Diagram for an Admin**

## Concept Page Layouts

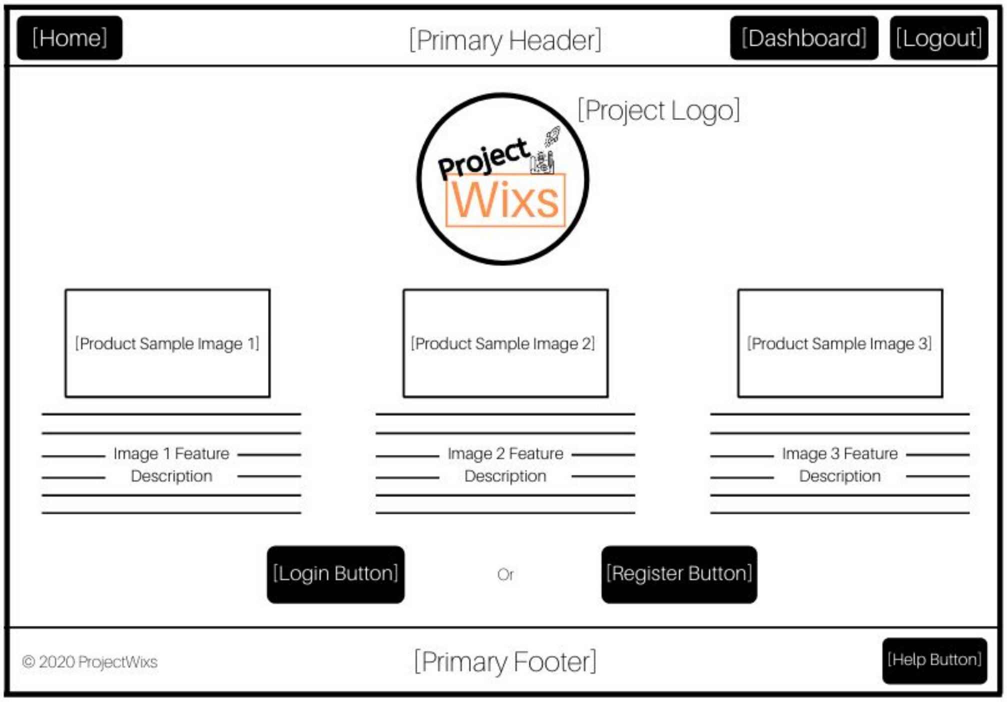


**Figure B-6: Concept Landing Page**

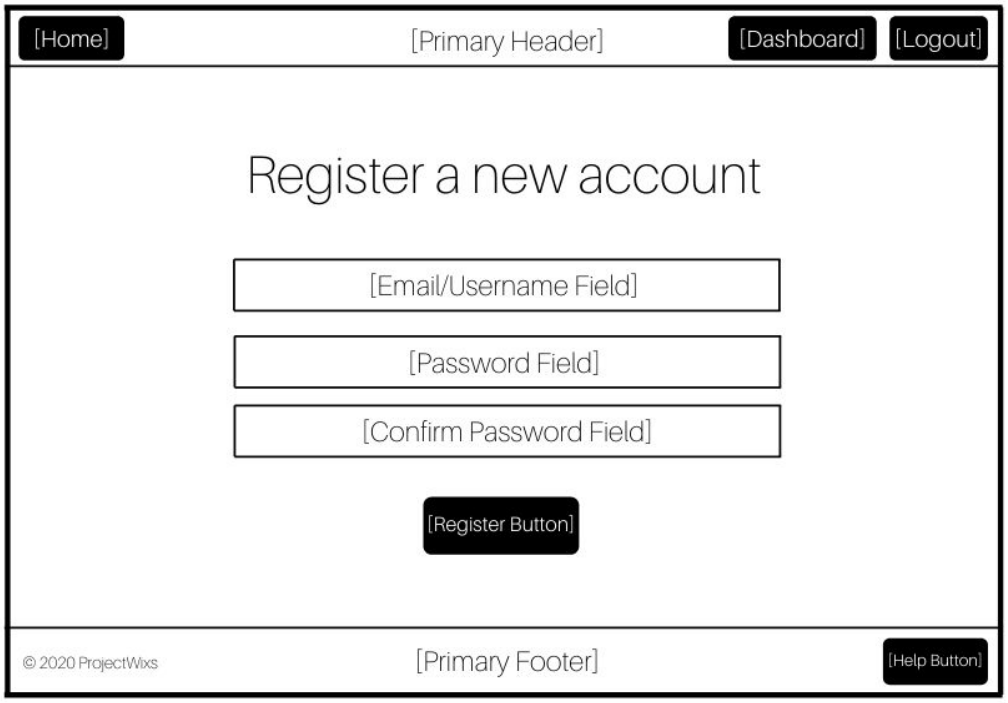


**Figure B-7: Concept Registration Page**

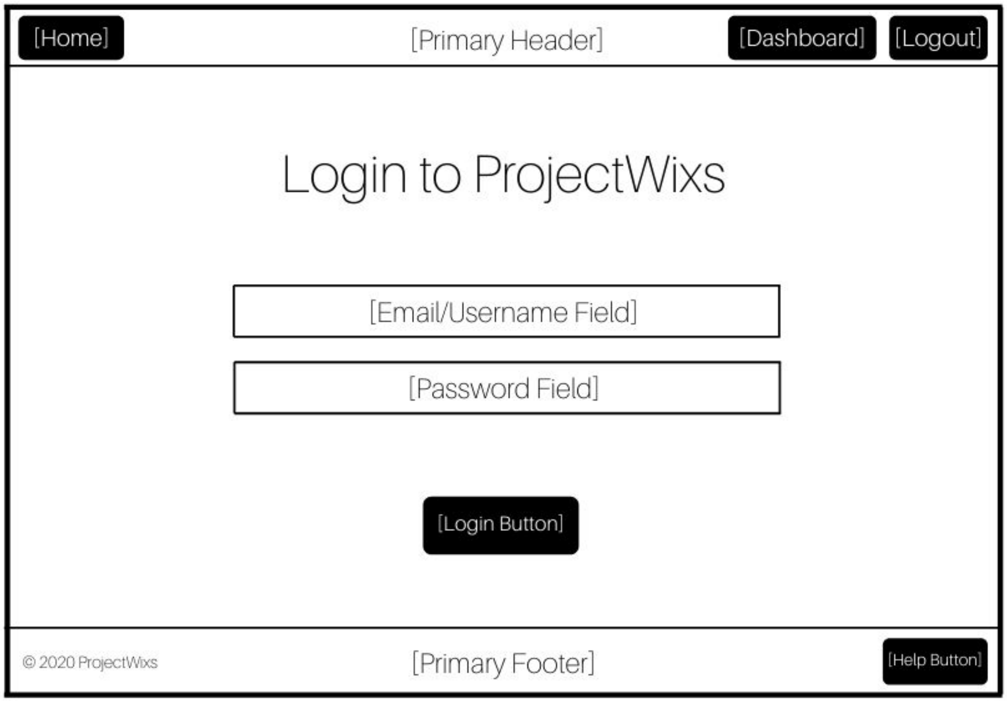


**Figure B-8: Concept Login Page**

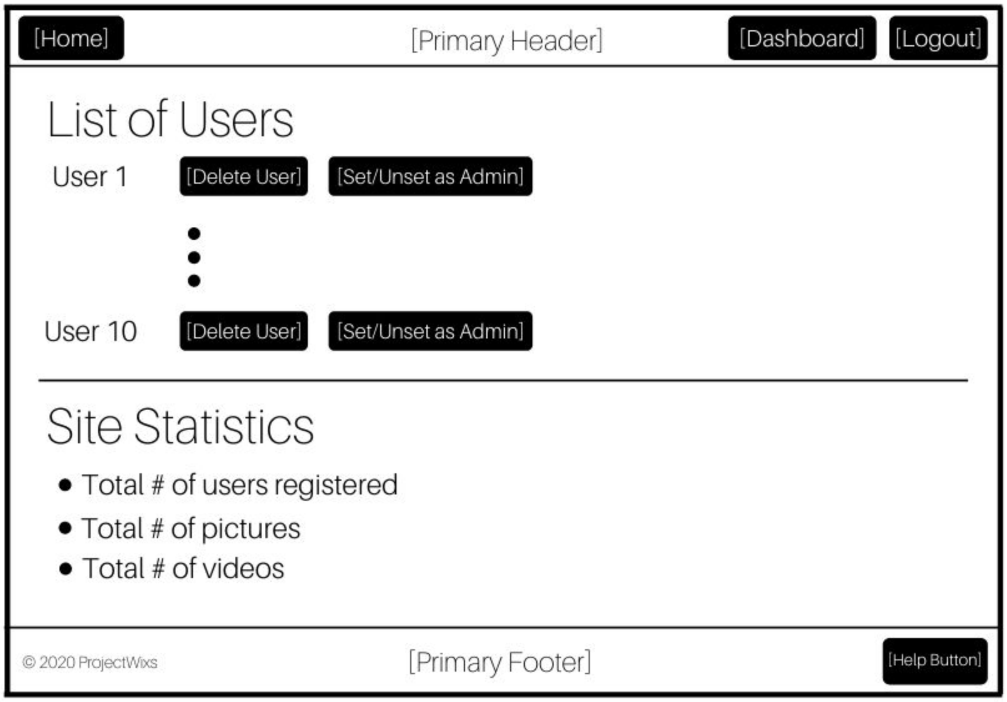


**Figure B-9: Concept Admin Tools Page**

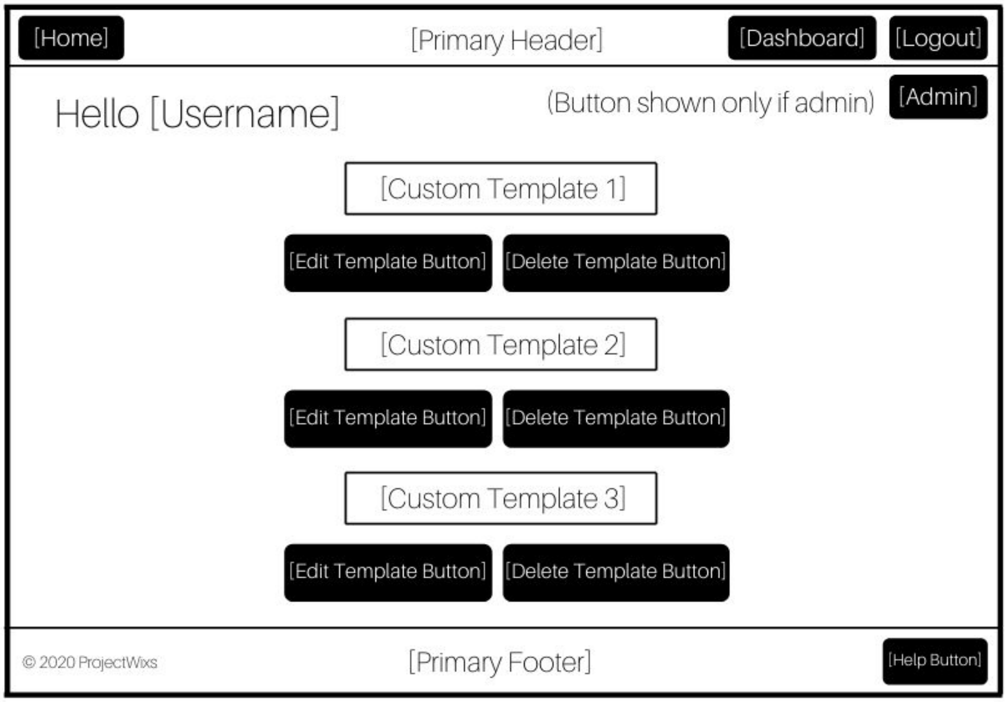


**Figure B-10: Concept Template Selection Page**

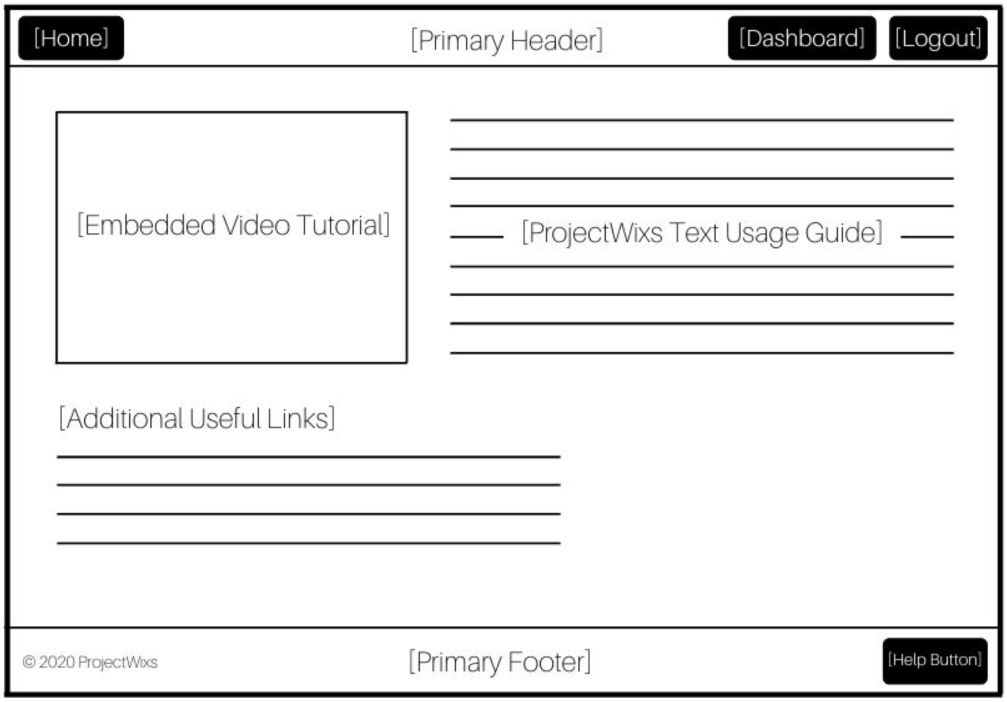


**Figure B-11: Concept Help Page**

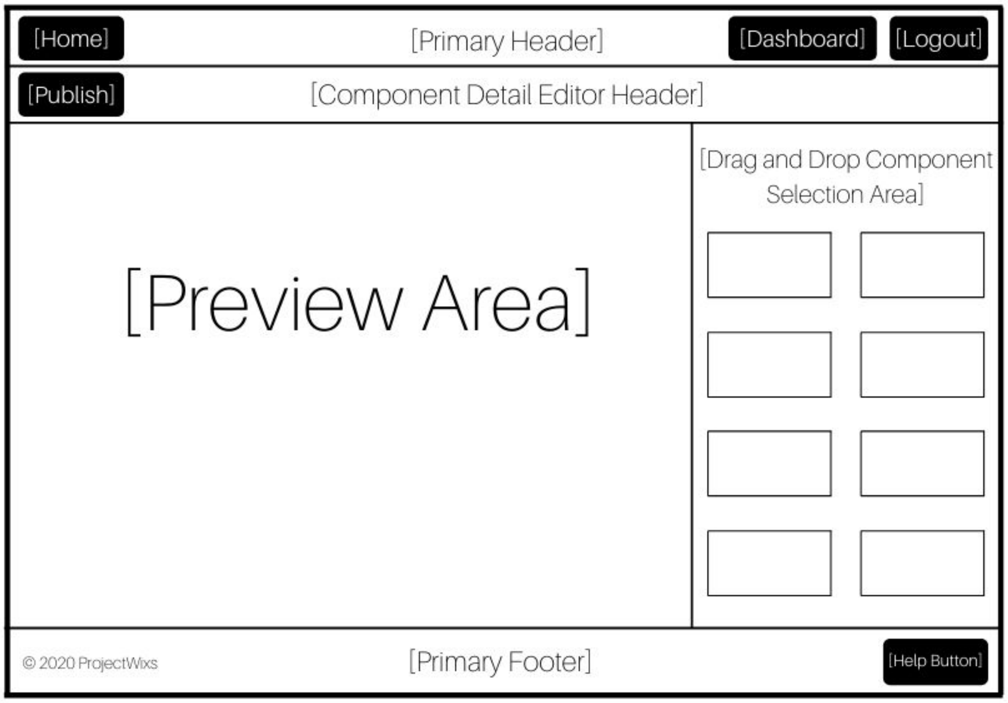


**Figure B-12: Concept Template Editor Page**

# Sequence Diagrams

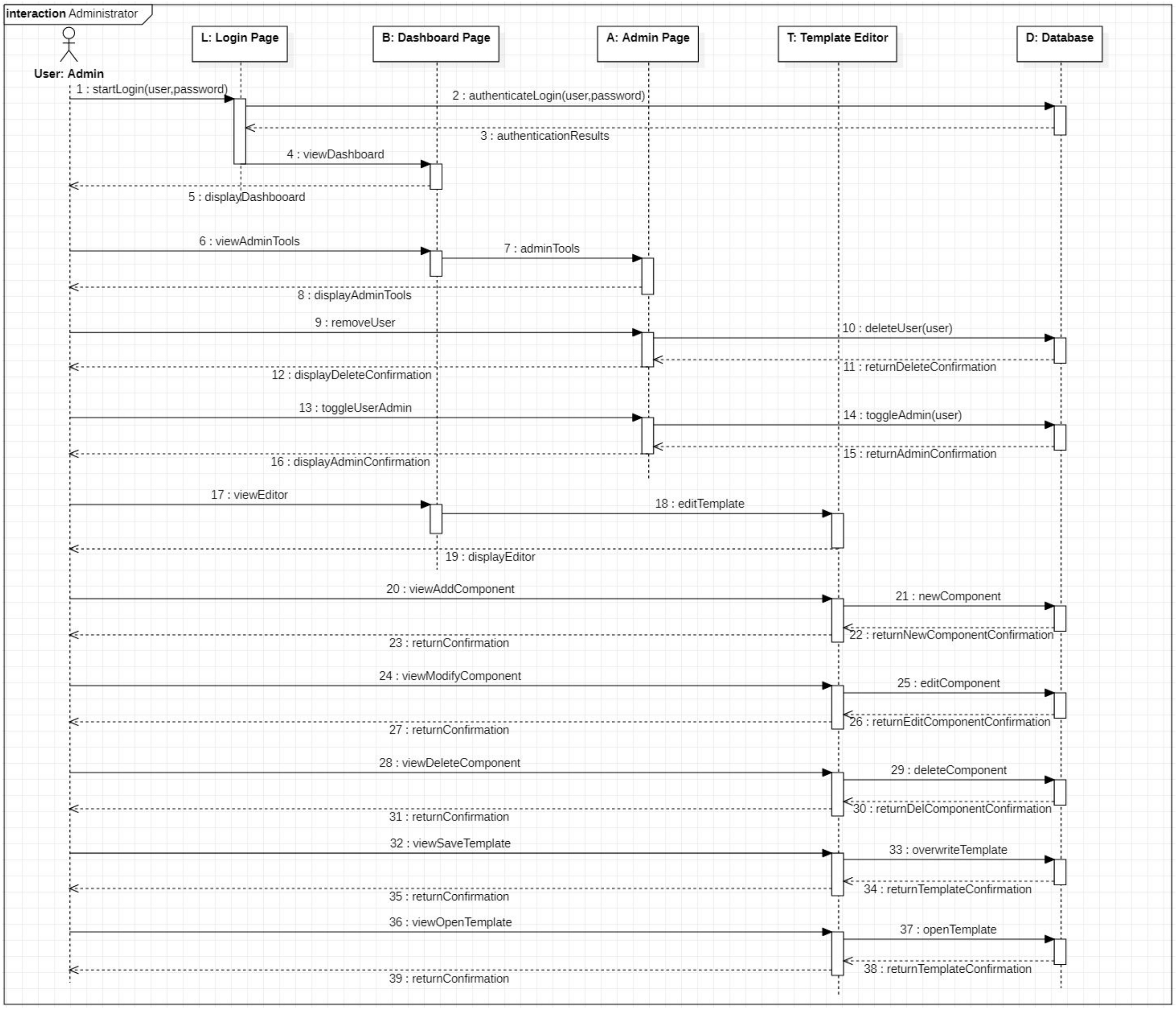


**Figure B-13: Administrator Sequence Diagram**

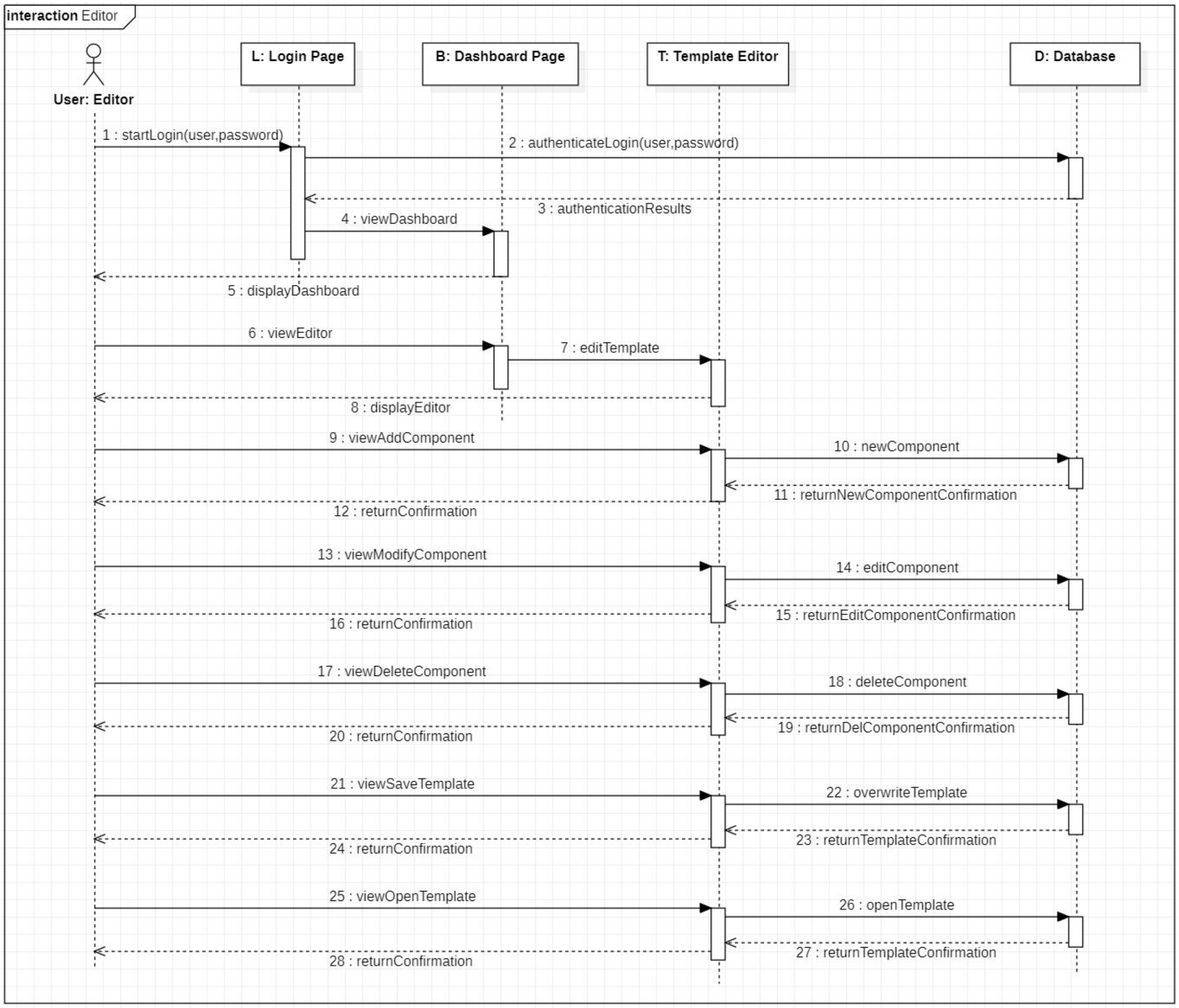


**Figure B-14: Editor Sequence Diagram**

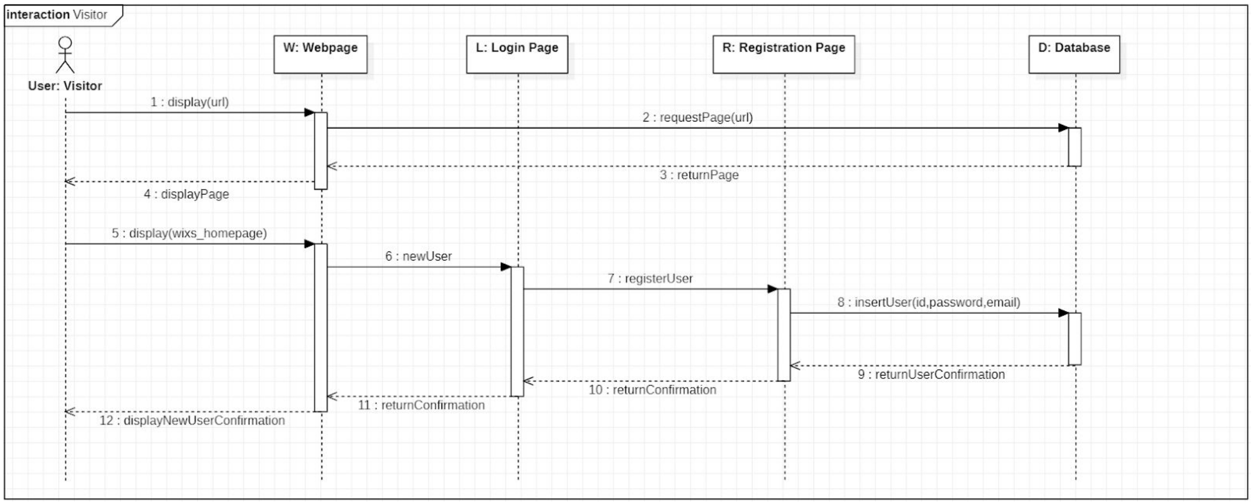


**Figure B-15: Visitor Sequence Diagram**

# Appendix C - Issues List

I. As of the current version, limits on accounts to be created and content upload file sizes are general guidelines without knowing accurate system limitations (without load stress testing). For now, the ProjectWixs team is limiting the numbers to what we feel are adequate early-stage limits for how many accounts can be registered and the size of files that can be uploaded.

II. As of the release of this document and without actual system implementation underway. Front-end frameworks to be used are currently tentative as the team finds an ideal solution. A framework will be decided upon which incorporates aforementioned system standards, style guides, and adaptability to changing goals.

III. Maximum timing goal of 1 minute for system performance requirements (See Section 5.1) is a general guideline and can likely be tweaked and reduced as system is fully implemented. It can act as a tentative overestimate.

IV. The referenced style guide is subject to change, depending on the team's personal preference.

V. Conceptual layout designs are subject to change as implementation takes place. For example, additional navigation buttons may be added to Figure B-8 in order to bring users to the registration page instead.

VI. Presently, without access to the group account, a temporary link to a referenced file hosted on a team member’s account is being used in this SRS.