

Static Site Generator

FINAL REPORT

Team 22

Client: Buildertrend

Project Manager: Gina Saccoman

Advisers: Samik Basu

Team Members:

Andrei Baechle

Joshua Flory

Bennett Ray

Luke Shay

Ty Trinh

Team Email: sdmay21-22@iastate.edu

Team Website: <https://sdmay21-22.sd.ece.iastate.edu/>

Revised: 04.25.2021

Executive Summary

Development Standards & Practices Used

Development Standards

- ISO/IEC 12207 - Software Life Cycle Processes: outlines the process for developing and maintaining software
- ISO/IEC 29148 - Requirements engineering: guidelines for applying requirements and requirement-related processes as described in ISO/IEC 12207

Software Practices

- Agile Development
- Unit Testing
- Branch-Review-Merge Workflow

Summary of Requirements

- Sites generated must be static
- Users will be given base templates to edit
- Sites will have different theme options
- Some content will be integrated with Buildertrend (login, lead generation, contact)
- Admin panel to edit templates
- Basic api to save and load edited templates
- Users will be able to view saved sites by visiting the site domain

Applicable Courses from Iowa State University Curriculum

- S E 319
- COM S 309
- S E 329
- S E 339

New Skills/Knowledge acquired that was not taught in courses

- React
- TinaCMS
- Express
- TypeScript

Table of Contents

| | |
|---|-----------|
| 1 Introduction | 7 |
| Acknowledgement | 7 |
| Problem and Project Statement | 7 |
| Operational Environment | 7 |
| Requirements | 7 |
| Intended Users and Uses | 8 |
| Assumptions and Limitations | 9 |
| Expected End Product and Deliverables | 9 |
| 2 Project Plan | 10 |
| 2.1 Task Decomposition | 10 |
| 2.2 Risks And Risk Management/Mitigation | 11 |
| 2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria | 12 |
| 2.4 Project Timeline/Schedule | 13 |
| 2.5 Project Tracking Procedures | 17 |
| 2.6 Personnel Effort Requirements | 18 |
| 2.7 Other Resource Requirements | 18 |
| 2.8 Financial Requirements | 18 |
| 3 Design | 18 |
| 3.1 Previous Work And Literature | 18 |
| 3.2 Design Thinking | 19 |
| 3.3 Proposed Design | 20 |
| 3.4 Technology Considerations | 20 |
| 3.5 Design Analysis | 22 |
| 3.6 Development Process | 22 |
| 3.7 Design Plan | 22 |
| 3.8 Physical Security | 23 |
| 3.9 Cyber Security | 24 |

| | |
|------------------------|----|
| 3.10 Design Evolution | 24 |
| 4 Testing | 25 |
| 4.1 Unit Testing | 25 |
| 4.2 Interface Testing | 26 |
| 4.3 Acceptance Testing | 26 |
| 4.4 Process | 26 |
| 4.5 Results | 27 |
| 5 Implementation | 29 |
| 5.1 File Server | 29 |
| 5.2 CMS Site | 30 |
| 6 Closing Material | 34 |
| 6.1 Conclusion | 34 |
| 6.2 References | 35 |
| A1. Operational Manual | 35 |
| A1.1 Setup | 35 |
| A1.2 Demo | 35 |
| A1.3 Testing | 35 |

List of figures/tables/symbols/definitions

Use Case Diagram: Diagram representing how different types of users will interact with our system.

Task Decomposition Tables: For each subtask associated with a task, lists the number of the subtask, name, and which subtasks need to be completed before the task.

Task Dependency Diagrams: Representation of all subtasks within each task and the order in which they need to be completed and which tasks can be completed concurrently.

Subtasks Lists: Lists the subtasks for each semester. Each subtask has a start date and expected duration in days indicating how long the subtask will take to complete.

Gantt Charts: The Gantt charts for Fall 2020 and Spring 2021 generated from the subtask lists.

Required Time Table: For each task, lists the task number, duration in days, and the expected number of hours that each task will take to complete.

Component Diagram: Component diagram representing the primary components in the project and how they connect.

System Block Diagram: A simple block diagram of our project.

Test Process Diagram: Process for working on tasks from beginning to end including when testing needs to be completed.

1 Introduction

1.1 ACKNOWLEDGEMENT

Amanda Dropinski: Project Manager (Semester 1)

Amanda was our project manager for the majority of the first semester of the project. She set up our initial and recurring meetings this semester and introduced us to this project.

Gina Saccoman: Project Manager

Gina Saccoman took over for Amanda for the remainder of the first semester and the second semester. She assisted with organizing check ins and understanding the project purpose and goals.

Thomas Judge: Application Developer

Thomas is a full-time application developer from Buildertrend who provided technical assistance for us during project development.

1.2 PROBLEM AND PROJECT STATEMENT

Problem Statement

Buildertrend wants to provide clients with a way to build and maintain their company websites.

Solution Approach

The Static Site Content Management system allows clients to easily set up and maintain a company website via an administrative panel and will provide Buildertrend's clients a custom website building experience so they can stand out in their respective markets.

1.3 OPERATIONAL ENVIRONMENT

There is no physical component to our project, and all work for our project was completed on our personal machines.

1.4 REQUIREMENTS

Functional Requirements

- Admin panel to edit basic JSON file that drives the site
- Generated sites must be static
- Users will be able to use generic templates for their sites including Home, Blank, Gallery,

- Showcase, Blog, Login, Contact
- Users will be able to choose from at least 2 themes
- Client sites will be integrated with Buildertrend login form
- Basic API to save and retrieve JSON data for generated sites

Technical Requirements

- Frontend application will be created using React with TypeScript
- TinaCMS will be used to build the Content Management System
- All rendering will be handled on the client side
- Node will be used to build the file server

Non-Functional Requirements

We were not given any explicit non-functional requirements, but we can assume there are some general non-functional requirements.

- Admin site will be able to support multiple users editing different sites.
- The file server will be easy to replace
- Generated sites will be able to support all of the client's customers
- Generated sites will work on both mobile and desktop

1.5 INTENDED USERS AND USES

Users

- Admin: The Buildertrend clients who are using the site to create their websites for their customers.
- Customer: The customer of the client who will be using the sites generated by the admins.

Use Cases

Admin

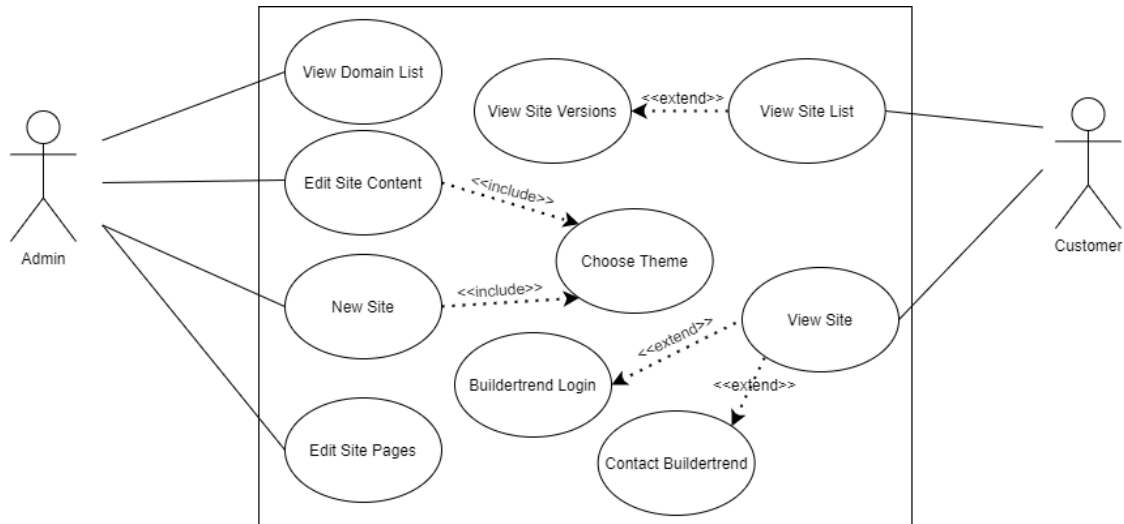
- View domain list: View domains they own
- Create a site: Create a new site by picking a domain
- Edit site content: Edit the content on the site pages
- Edit site pages: Choose which templates to include in the site
- Choose site theme: Choose the site-wide theme

User

- View Site List: View a list of sites they can view
- View Site Versions: View the different versions of each site
- Contact Buildertrend: If the site includes the contact template, they can use the contact form to contact Buildertrend.
- Buildertrend Login: If the site includes the login template, they can use the form to log in to their Buildertrend account.

Use Case Diagram

This diagram shows how the different users will interact with our application and how some use cases are connected.



1.6 ASSUMPTIONS AND LIMITATIONS

Assumptions

- We only need to support english because all of their clients are in US
- There will be no unexpected costs since we have no budget
- All planned Buildertrend integrations will have no alternative integrations
- Our NodeJS server does not need to be very in depth because it will be replaced

Limitations

- Both semesters are shorter than previous semesters, so we had fewer weeks to complete our milestones
- A full time team will be taking over the project at the end of May, so we are expected to only use frameworks and libraries that Buildertrend uses
- During this semester, we will not be performing tests for high volume traffic on client websites or the API endpoints
- The output sites do not have server-side rendering, so that means that we have to have to make these sites static

1.7 EXPECTED END PRODUCT AND DELIVERABLES

End Product: Static Site Content Management System

Deliverable 1: File Server

Delivery Date: 10.13.2020

This deliverable is for the basic file server that will store and return the edited files for the sites created by the users. This component will be relatively simple since it will be replaced after the project is given to Buildertrend.

Deliverable 2: Static Site CMS Web Application.

Delivery Date: 04.20.2020

This deliverable is for the frontend React application that users will use to create their websites. Buildertrend's customers will be able to use the admin template to edit provided templates to build their sites. Sample templates will include, home, gallery, login, blank, showcase, blog and contact. Each template will have sample content that users can edit to fit their site. Some features will be integrated with the Buildertrend site, like the login and templates with lead generation. Admins will also have the options to choose different themes for their sites. Once the admin has completed the site, their customers will be able to visit the domain for the site to view it.

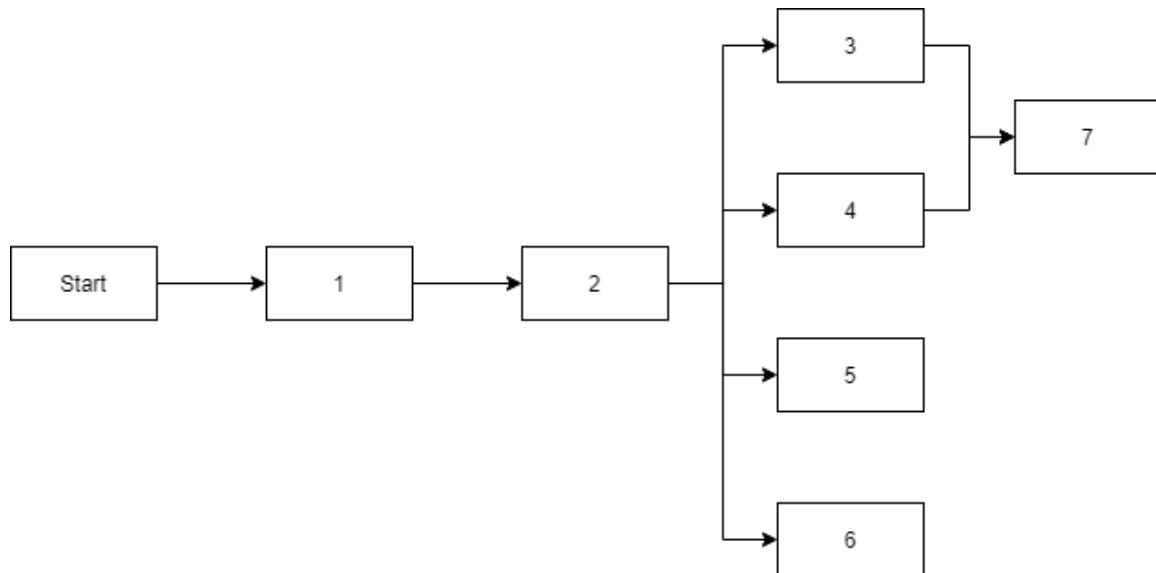
2 Project Plan

2.1 TASK DECOMPOSITION

Tasks

1. File Server
2. Admin Panel and Site Viewing
3. Template Creation
 - a. Home
 - b. Gallery
 - c. Showcase
 - d. Blog
 - e. Login
 - f. Contact
4. Sample Content
 - a. About
 - b. Floor plan
 - c. Testimonials
 - d. Services
 - e. Privacy Policy
 - f. Process
5. Themes
 - a. Light
 - b. Dark
 - c. Minimal
6. Site Versioning
7. SEO Optimization

Task Dependencies



2.2 RISKS AND RISK MANAGEMENT/MITIGATION

Risk Factor

Our risk factor scale is from 0 to 1, with 0 indicating that the potential risk is unlikely to be an issue, while 1 indicates that we expect to run into the issue.

Task 1: File Server

1. File permissions

Risk Factor: 0.1

Explanation: As of right now, we are using our own file server which will have the risk of having other users access files they should not have access to. Since our file system server is temporary, this has little risk to our implementation for the frontend.

Task 2: Admin Panel

1. TinaCMS does not fulfill requirements for our content management system.

Risk Factor: 0.2

Explanation: This library was chosen by Buildertrend who already tested the library out and determined it was the best CMS for the site. However, it is possible that as the site features

become more advanced, TinaCMS will prevent us from implementing features as originally anticipated.

Task 3: Template Creation

1. Templates mockups may change as project progresses

Risk Factor: 0.1

Explanation: As templates are completed, Buildertrend may revise what templates need to look like. This is not a big risk, but this may be in any capacity, so we should be prepared to change.

2. Cross-Browser Compatibility

Risk Factor: 0.4

Explanation: Different web browsers interpret web pages in different ways, so some browsers may not support certain stylings or functionality. In order to keep these pages consistent, we will have to be aware of these differences. We know that people still use Internet Explorer or outdated versions of other browsers, so we expect this to be an issue. For example, Internet Explorer does not always support some functionality found in Bootstrap which we will be using for our UI.

2.3 PROJECT PROPOSED MILESTONES, METRICS, AND EVALUATION CRITERIA

Milestone 1

Title: Basic File Server and Frontend Application

Due Date: 10.13.2020

Evaluation Criteria: Users will be able to run both the file server and frontend application and will be able to save a JSON body to the file system and download a file.

Milestone 2

Title: Admin Panel

Due Date: 10.27.2020

Evaluation Criteria: Users will be able to use the admin panel to edit and save a template file. The user will then be able to view the edited template.

Milestone 3

Title: 3 templates and one integration

Due Date: 11.10.2020

Evaluation Criteria: Users will be able to build sites by editing the templates. One of the templates will include integration

Milestone 4

Title: Remaining Templates and 2 themes

Due Date: 02.23.2021

Evaluation Criteria: Admins will be able to create sites using any of the templates that were planned for implementation. They will also be able to choose between 2 different themes that will be applied throughout their site.

Milestone 5

Title: 3 Pages, 1 theme, and Google Maps integration.

Due Date: 03.23.2021

Evaluation Criteria: Admins will be able to add 3 of the following pages: About, Floor Plan, Testimonials, Services, Privacy Policy, Process to their site. They will also be able to choose another theme for their site as well. The contact us form should also include an integration with Google Maps.

Milestone 6

Title: Remaining pages and BT Reviews Integration.

Due Date: 04.20.2021

Evaluation Criteria: Admins will be able to add the remaining pages that have not been implemented by this point. They will also be able to choose to create a page using the Reviews template which will feature Buildertrend reviews.

2.4 PROJECT TIMELINE/SCHEDULE

This subtask table includes how long we estimated each task would take to complete. We first determined how many days we thought each task would take and then determined what the start

date should be to make sure that we completed the task before the milestone for that task. Both tables were used to create the Gantt chart.

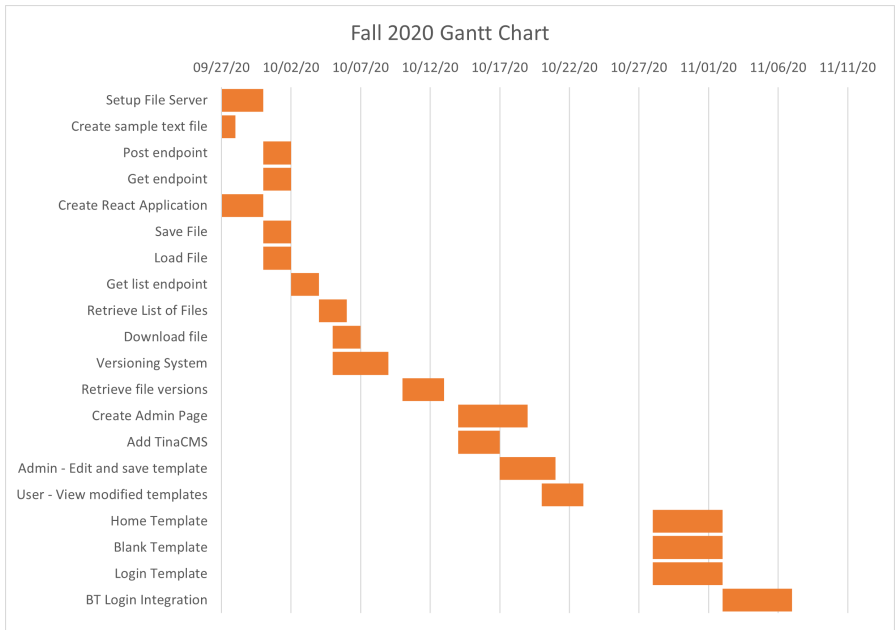
Subtask Table

| Semester 1 | | |
|--------------------------------|------------|-----------------|
| Task | Start Date | Duration (Days) |
| Setup File Server | 09/27/20 | 3 |
| Create sample text file | 09/27/20 | 1 |
| Post endpoint | 09/30/20 | 2 |
| Get endpoint | 09/30/20 | 2 |
| Create React Application | 09/27/20 | 3 |
| Save File | 09/30/20 | 2 |
| Load File | 09/30/20 | 2 |
| Get list endpoint | 10/02/20 | 2 |
| Retrieve List of Files | 10/04/20 | 2 |
| Download file | 10/05/20 | 2 |
| Versioning System | 10/05/20 | 4 |
| Retrieve file versions | 10/10/20 | 3 |
| Create Admin Page | 10/14/20 | 5 |
| Add TinaCMS | 10/14/20 | 3 |
| Admin - Edit and save template | 10/17/20 | 4 |
| User - View modified templates | 10/20/20 | 3 |
| Home Template | 10/28/20 | 5 |
| Blank Template | 10/28/20 | 5 |
| Login Template | 10/28/20 | 5 |
| BT Login Integration | 11/02/20 | 5 |

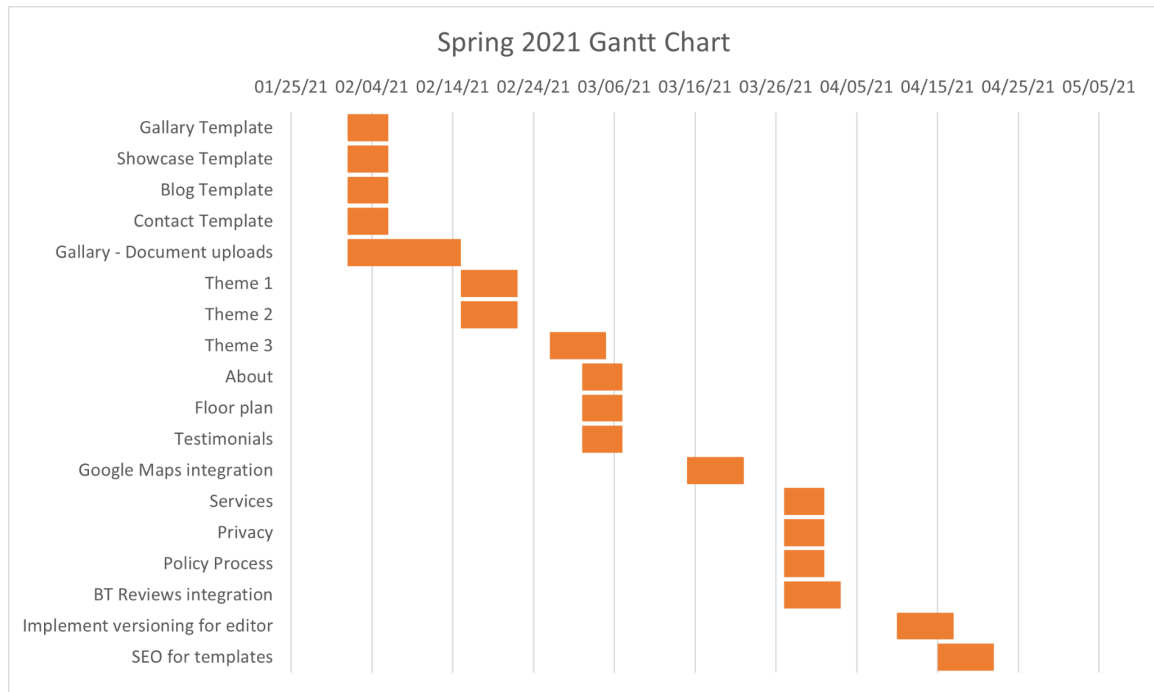
| Semester 2 | | |
|----------------------------|------------|-----------------|
| Task | Start Date | Duration (Days) |
| Gallery Template | 02/01/21 | 5 |
| Showcase Template | 02/01/21 | 5 |
| Blog Template | 02/01/21 | 5 |
| Contact Template | 02/01/21 | 5 |
| Gallery - Document uploads | 02/01/21 | 14 |
| Theme 1 | 02/15/21 | 7 |
| Theme 2 | 02/15/21 | 7 |
| Theme 3 | 02/26/21 | 7 |
| About | 03/02/21 | 5 |
| Floor plan | 03/02/21 | 5 |

| | | |
|---------------------------------|----------|---|
| Testimonials | 03/02/21 | 5 |
| Google Maps integration | 03/02/21 | 7 |
| Services | 03/27/21 | 5 |
| Privacy | 03/27/21 | 5 |
| Policy Process | 03/21/21 | 5 |
| BT Reviews integration | 03/27/21 | 7 |
| Implement versioning for editor | 04/10/21 | 7 |
| SEO for templates | 04/15/21 | 7 |

Semester 1



Semester 2



2.5 PROJECT TRACKING PROCEDURES

Slack

Our team used slack for communication. We have a dedicated channel where we discussed the updates to the project. This was also a means to communicate between team members about questions regarding the implementation of a specific pull request.

Github Project

Our team used the project feature in Github to track the progress of the task. Each task was represented as an issue on the project board, and team members could assign themselves to the issue when they picked up an issue.

Project Board Columns

To-Do: Cards in this column have not been started but are ready to be picked up by a developer

In Progress: Column for cards that have been started but no pull request has been created

Code Review: Column for cards that have pull requests that need to be reviewed

Testing: Cards whose functionality has been merged to master but a developer still needs to verify the functionality still works.

Done: Card works in master and is complete.

2.6 PERSONNEL EFFORT REQUIREMENTS

Time Required for Each Task

To determine the number of hours required for each task, multiplied the duration from the table in 2.4 by 2, since we assumed that we would spend, on average, 2 hours a day working on each subtask.

Required Time Table

| Task | Duration (Days) | Hours | Explanation |
|------|-----------------|-------|---|
| 1 | 28 | 56 | Task 1 involves a lot of setting up components that will be used for the rest of the project. We assumed it would take some time to establish the development cycle during the first task. |
| 2 | 24 | 48 | Task 2 has fewer subtasks than task 1, but the subtasks are more complex than the ones in task 1. Since the due date for this milestone will most likely need to be moved up, we expect to spend less time on this task. |
| 3 | 20 | 40 | Task 3 also has fewer subtasks than task 1, and we believe that the subtasks will be relatively easy to implement. Like task 2, we will also have to move up the due date. |
| 4 | 36 | 72 | Task 4 will require more time due to the number of templates and the 2 themes that will need to be implemented. We expect the themes to take a while to implement across all of the templates. |
| 5 | 29 | 58 | Task 5 includes the addition of more content to the templates, another theme, and google maps integration. We expect that adding a 3rd theme and implementing the google maps integration will be the most time consuming subtasks for this task. |
| 6 | 36 | 72 | Task 6 involves implementing the remaining features for this project. This task has some potentially complex subtasks like the reviews integration and SEO, so we anticipate having to spend a significant amount of time on this task. |

Total # of Hours: 346

Personal Effort Estimation

Since there are about 23 weeks left in the semester, and we have 6 group members, we expect the personal effort to be about 2.5 hours per week for project implementation.

2.7 OTHER RESOURCE REQUIREMENTS

Github will be used for source control for the project.

2.8 FINANCIAL REQUIREMENTS

Financial requirements are not relevant for this project since there are no anticipated costs.

3 Design

3.1 PREVIOUS WORK AND LITERATURE

The design of our static site generator project mirrors tools and functionalities implemented in other popular website building services. This section will look at competing website generators; comparing and contrasting their features in order to learn what a user would look for in their ideal site building experience.

Squarespace

Squarespace provides software as a service for building and hosting websites. They provide templates for different types of websites including photography, blogs, and online stores. Plans cost \$12-\$46 a month.

Advantages:

- A lot of templates and customization options.
- Allows coding and customization through CSS, HTML, JavaScript.
- Drag and drop user interface.
- Community support through forums and guides.
- Provides Google Analytics [2].
- Fully responsive templates.

Shortcomings:

- Squarespace won't help with troubleshooting if CSS, HTML, or JavaScript has been manually changed [3].
- Restricting content through membership on a site can be difficult [3].
- User interface and code changes frequently [3].

Wix

Wix is another website that allows users to build their own websites using templates. The basic plan Wix plan is free, but has limited features. Premium plans cost \$14-\$39 a month.

Advantages:

- More beginner-friendly than Squarespace [2].

- Drag and drop user interface.
- Free basic plan.
- Provides Google Analytics [2].

Shortcomings:

- Templates are not responsive [1] .
- Cannot use CSS, HTML, or JavaScript.
- Templates cannot be changed after the website has been built [1].

Our Project: Static Site Generator

Our project features some of the functionality from both Squarespace and Wix, but given the time constraints our site is more limited than either of those options. The goal of this project is not to compete with other website building platforms, but to provide added value in being a Buildertrend customer.

Advantages:

- The intended users for the site will be more specific than Squarespace's and Wix's intended users, allowing us to create an experience that is more tailored to Buildertrend's customers.
- Less features should make it easier and faster to create sites.

Shortcomings:

- Limited development time will make it difficult to implement a lot of the features that Wix and Squarespace have.
- Templates allow limited customization
- Few themes will be provided to users

3.2 DESIGN THINKING

Define Aspects

- Buildertrend customers need to be able to market themselves and their products to their customers.
- Buildertrend customers do not have a way to create their own sites without using sites like Squarespace or Wix or hiring a developer to create it for them.
- Buildertrend customers need something that is easy to navigate and does not take a long time to learn to deploy a website.

Design Choices (already provided by Buildertrend)

- Creating a file server to save templates and edited site content.
- Creating a frontend React with TypeScript application for building the sites.
- Using TinaCMS to implement the content management system.

3.3 PROPOSED DESIGN

File Server

Since the file server will be replaced once we give the project to Buildertrend, we had some more freedom to choose the implementation approach. However, they suggested we use Node since it was lightweight and already had a file system module for creating and modifying files on our local machines. We decided to use Node with Express to create the rest api with endpoints for saving and retrieving files along with additional endpoints for working with images.

Web Application

The web application will be a React with TypeScript application as specified by our client. Using React with TypeScript is standard for all new web applications created by Buildertrend, so we had to follow that standard. The decision to use TinaCMS to build the content management system was already provided by Buildertrend who determined that it would be the best editing toolkit for our project.

Even though we did not experiment with other approaches, Buildertrend had already created a test application to ensure that TinaCMS had the features needed to fulfill the requirements for our project. It allows users to edit provided templates and save those changes, and then we can save those modified files to our file servers. Most of our work will involve creating the default templates and the content for the templates.

3.4 TECHNOLOGY CONSIDERATIONS

React with TypeScript

Strengths

- Large development community.
- There are many React tutorials available.
- Already used by Buildertrend, so they can help us with development.
- TypeScript is more familiar to people who have experience with Object-oriented programming, as opposed to JavaScript which is a scripting language.
- TinaCMS is designed to work with React.

Weaknesses

- React is not taught in any course at ISU, so for people who have not used it before there may be a learning curve.
- Most React tutorials use JavaScript and not TypeScript, so it can be harder to find tutorials or help online that use React with TypeScript.
- React creates single page applications which are not as performance oriented as other options.

Alternatives

- **Angular:** Another web application framework that could be used to create the web application. Angular already uses TypeScript by default, so it would be easier to find documentation and guides that use TypeScript. However, we will be giving this project to Buildertrend when we are done, and Buildertrend does not use Angular.
- **Vue with TypeScript:** A relatively new web application framework that could be used instead of React with TypeScript. Vue supports using Typescript, but again, Buildertrend uses React for web development.

Node

Strengths

- Lightweight, don't need to create a large application to create a simple file server
- Easy to learn
- Already has modules for working with a device's file system
- Large package ecosystem

Weaknesses

- Relatively simple; if we end up needing to put more work into the file server, we may want to re-evaluate using Node.
- Not secure compared to other runtimes.

Alternatives

- **Creating a backend application with a database:** If we decide that the file server needs to be more complex than initially specified, we could replace the Node file server with a backend application that is connected to a database. However, we do not anticipate having to work on the file server any more than we already have, so this will not be necessary.

TinaCMS

Strengths

- Allows editing of templates.
- Easy to add to a preexisting project.
- Allows for generation of static websites.

Weaknesses

- New technology to the team; this may be challenging to work with.

Alternatives

- **Magnolia CMS:** Another content management system toolkit that works with React. Allows a lot more customization and features than TinaCMS, but is not free.

3.5 DESIGN ANALYSIS

File Server

We know that using Node for the file server works since we have already completed and demoed the file server to our client. This component was intended to be relatively simple so we most likely will not have to modify it again.

Web Application

Based on the test application that Buildertrend created that was using TinaCMS, we know that the design will work for our project. However, that test application was a relatively simple application, so as we get farther into the project, we may realize that TinaCMS does not have all of the necessary features to fulfill the project requirements. The test application also did not feature saving modified files, so we will have to figure out how to implement that aspect of our project as well.

3.6 DEVELOPMENT PROCESS

For our development process, we will be following the Kanban system for development. We decided that since we are all working from home and it is difficult to find meeting times where all team members could be present, the best development process would be Kanban. Since we are using Github for source control, we are using the project feature in Github to track our progress and as our Kanban board. Team members will pick up cards that have assigned issues and will move the cards between columns as work on the issue.

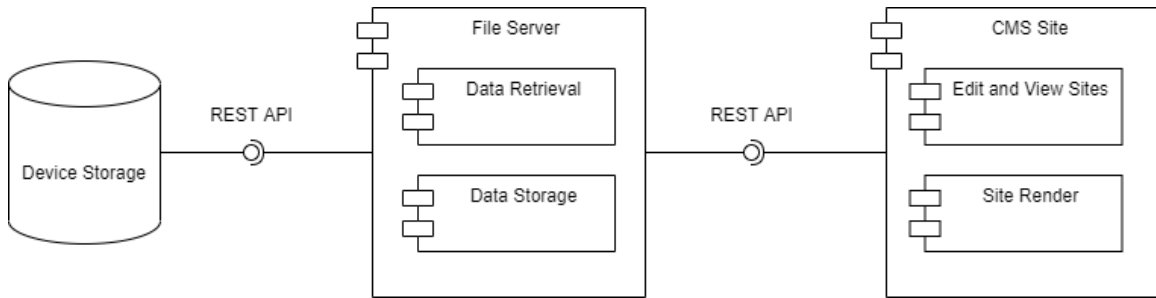
For our git workflow, as a team member picks up a card, they will create a branch to work on the feature and request a pull request. Once the feature is implemented by the team member, there will be a code review by the other team members. Once the code review is completed, the pull request will be accepted and merged into the master branch.

3.7 DESIGN PLAN

Component Diagram

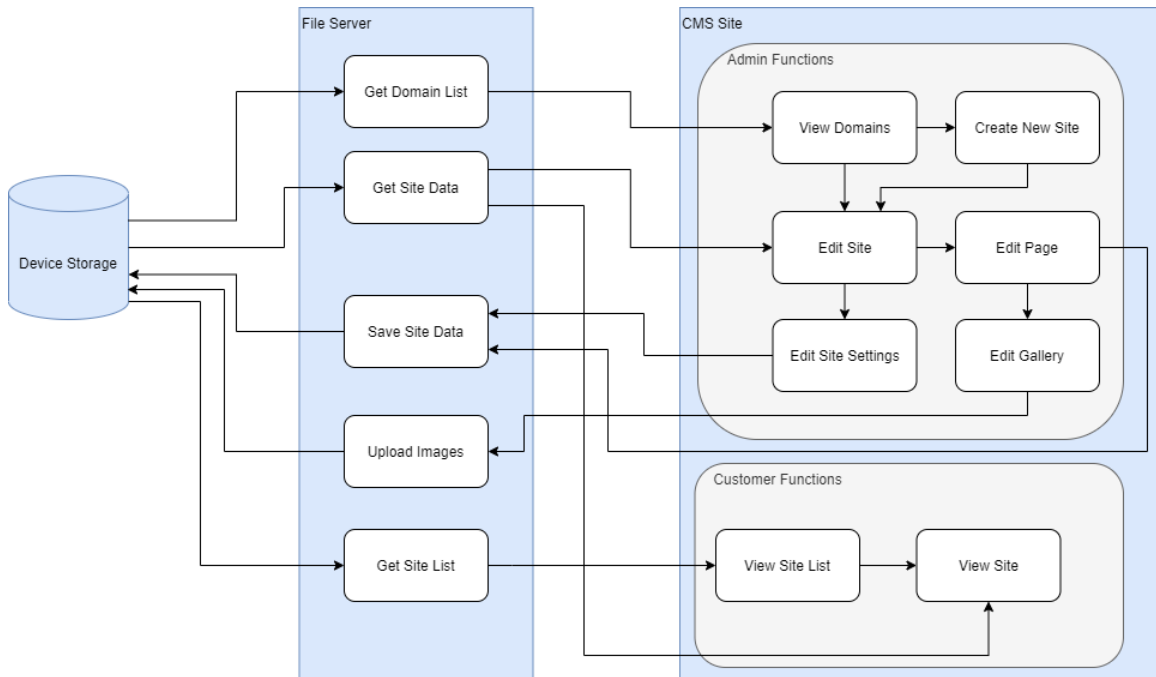
This diagram represents the main components for this project. The main components include:

- View and Edit sites: Represents how the users will interact with the generated sites
- Site and Page render: How the frontend application will handle rendering the json data received from the file server
- Data retrieval: Endpoints for retrieving data on the file server
- Data storage: Endpoints for saving data to the file server



System Block Diagram

This diagram represents the communication between the file server and CMS site. The nodes on the file server represent the get and post endpoints, while the nodes on the CMS site represent the different pages and key functionality of the site.



3.8 PHYSICAL SECURITY

The only physical component to our project is the computers that we used during development. Our project did not have a physical component, so the physical security of the project itself is not a concern. Since the code and repository are property of Buildertrend, we had to complete Non-Disclosure Agreements in order to be added to the repository. We have access to the code on our personal computers, and if someone else were to access one of our computers, they would have access to the code and potentially steal it. We did not handle deployment of the project to a server, so the physical security of servers was also not a concern.

3.9 CYBER SECURITY

For the cyber security of our project, an important aspect to consider is the security of the tools we are using. Both our client and our team had not initially thought of the potential security concerns of any of the tools we were using during our design process. However it would be a good idea to examine the potential security risks once it is handed over to Buildertrend.

Tool Vulnerabilities

One security concern that we could have looked into was potential vulnerabilities associated with the tools we were using. We could have checked vulnerability databases like MITRE's CVE database or Rapid 7's Vulnerability or Exploit database. We could have also used free vulnerability scanners like Grabber or Vega to conduct pen testing against our site. If we found vulnerabilities we would implement the suggested countermeasures including updating libraries or using alternative more secure tools if possible. Since Buildertrend will be taking the project over after the semester is over, it is possible that they will perform their own analysis with tools they have, but at that point we will no longer have access to the project and the security will be their concern.

User Input Validation

Another cyber security concern that we had not considered was handling user input. Our front end application involves a considerable amount of user input. Most of the input is saved as a string to a file, so there shouldn't be any issues with cross-site scripting or injection attacks, but it would not have hurt to perform input sanitation before saving the input to a file. We could have also made sure to perform input sanitation any input from when the user creates or edits a site. Again, most of those inputs are saved to a file or used to create folders and it would be difficult to use the input to execute malicious code or modify something on the file server.

Our application does feature several iframes that access Buildertrend's systems and allow user input, however the security for those systems is not within the scope of the project.

3.10 DESIGN EVOLUTION

Since we had already started implementation during our first semester. Our design did not change significantly between semesters. We had already set up both the file server and CMS application, and most of the work completed this semester involved adding more templates, pages, and site settings. Most of the templates and pages were able to be added without adding any new features to either the file server or React application. For new templates and pages, we just had to create components, add routing for the component, and add the ability to add the page to the site from the new site page and settings modal. The additional site settings like the themes and accent color were also just added to the site page and settings modal. The only design change was the addition of image uploading for the gallery template.

Image Uploading for the Gallery Template

Image uploading on the frontend was handled by TinaCMS's built-in media manager and was included in the inline editing and forms. For all pages except the gallery page, the images were saved on site file. For adding images to a gallery, we used an HTML form, and saved each image to a gallery folder on the backend. This was the only endpoint added that was not pulling data from the site file or accessing the list of sites that we added since we had initially assumed we could handle the image uploading for the gallery through TinaCMS.

4 Testing

4.1 UNIT TESTING

File Server

The logic behind the endpoints for our file server will need unit testing. We will need to make sure that files are being saved and retrieved correctly from the file system. We will also use tests to ensure that the versioning system is working correctly. These tests also need to be in place for regression testing as changes are made to the file server logic. This will be done with Supertest and Jest.

React Application Unit Tests

The React Application will need unit tests as well. Buildertrend uses Jest and Enzyme for their React unit tests, so we will most likely use Jest and Enzyme for our project. This will ensure our pages render the information correctly and take in user input correctly.

Storybook

Buildertrend uses Storybook to test components in isolation. Storybook allows developers to build components without depending on other components or data from the file server. This can be used to test how components will look depending on what data is passed to it without having to save the data to the server.

Automated UI Testing - Selenium

We may look into doing automated UI testing with Selenium to test user navigation and interaction with the site. The site admins will have the ability to edit content on each page, which would require a lot of manual testing. Automated testing should ensure that all functionality in our site is working properly.

4.2 INTERFACE TESTING

Interface between File Server and React App

The only interface testing required will be between our file server and the React Application. To test retrieving from and saving files to our file server, we will be using Postman. Since our file server takes json content and saves it to a file, we just need to send a post request with sample json content in the body to test that the file is created and saved. We can then use get requests to test that the file was saved with the correct format and name.

We are not planning on testing the React Application by itself without already having the file server working properly. This would have required implementing a file system on the front-end side of the project as well, and we decided that the benefits of doing things would not be enough to justify the amount of work required to implement it. As we are connecting to the endpoints in the web application, we will have to make sure that we are getting and saving the files manually.

4.3 ACCEPTANCE TESTING

Acceptance testing for this project will take the form of 3 milestone demos each semester. During these demos, we will present our work to our project manager, full time developer, and members of the Buildertrend architecture team. If they are satisfied with our implementation of the milestone, we will be able to begin work on the next milestone. If they find issues with it or think that our implementation has not fulfilled the requirements, we will have to spend time revising our implementation and demo again during the next demo.

4.4 PROCESS

Process Outline

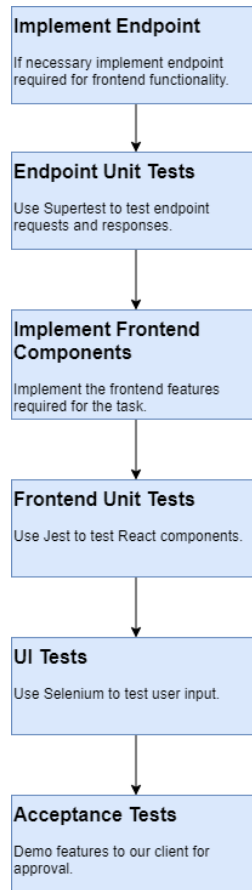
Our plan for testing throughout the project was to complete unit testing for each endpoint on the backend and each component on the frontend. Since most of the team was unfamiliar with React we decided that unit tests would be written after implementation and we would try to reach at least 85% code coverage. Since most of us had little to no experience testing React components, we were not sure how feasible this would be but wanted to at least have a goal for code coverage. We also planned on using Selenium to conduct UI testing once the features since there is a heavy emphasis on user interaction with the application. We did not plan on creating tests for the interaction between the file server and the React application since the file server is going to be replaced by Buildertrend and since we initially thought the file server would only be used for saving and retrieving the site files.

Our goal was to make sure that both the backend and frontend of the project would be thoroughly tested before demoing to our client to avoid running into technical issues or bugs during our demos and so the client would only have to focus on evaluating the features of the project. We also hoped that by creating unit tests and UI tests we would avoid issues with regression. Many of our tasks like

the template creation and sample content were not dependent on each other, so we did not want to have to go back and check that previously implemented features were still working properly. Having the tests in place would make sure that we did not accidentally break features that were previously working without us having to manually check each feature.

Process Diagram

For each task we wanted to follow the same process to ensure that testing was completed before demoing to our client.



4.5 RESULTS

Following the Test Process

During implementation we did not follow the test process we had developed. During work on the first milestone, we wrote tests for the api endpoints and some basic setup tests for the admin panel, but after that we did not write any other tests due to time constraints. After milestone 1, the tasks were more challenging than we had initially thought and as a result tasks were completed so close to the demo date that we did not have time to go back and create tests for the components. We also

ended up adding more endpoints but the implementation of the endpoints often changed as the implementation on the front end was reworked. We often had to prioritize making sure the features could be demoed instead of taking the time to write tests that could have saved us time later on.

One of the problems that we ran into early on was that the team did not have much experience with Jest and testing React applications. When we created the basic setup tests the admin panel was relatively simple and did not require much testing. As we added more functionality to the admin panel, we should have learned how to test the components as they became more complex instead of coming back to write tests after most of the functionality was completed. We did not have the time to learn how to test components that were as complex as the components we had at that point and as a result we did not write any more tests.

Impact

Some of the issues that we wanted to avoid by writing unit and UI tests came up during our implementation. For example, we completed the showcase, blog, and gallery templates for milestone 4 and then didn't need to work on these templates after that point. However, during milestone 5 created by working on a different feature created a bug that caused the templates to break. We did not notice the error until right before our demo and still don't know what caused the error. If we had the tests for those templates, we would have been able to determine exactly when the error occurred which would have narrowed down the possible causes of the problem.

Milestone 1 Acceptance Tests

We have already demoed the file server and basic file server to web application communication to our client. We were successful in meeting the requirements for the first milestone and even meeting some requirements for the next milestone. Our file server was able to save files after sample content was passed from the web application, and we were then able to retrieve that file from the file server on the web application. It was not required for the web application to be a React with TypeScript application, but we had decided to get ahead and create the application using React and TypeScript. The only other thing they mentioned was that they would not have implemented the versioning system the way that we did, but our implementation still worked.

Through this demo, we gained a better understanding of what our client was expecting for this project and demonstrated that we understood the project requirements. Since we most likely will not have to revise the file server, we do not expect significant redesigns of the file server, but we may revise the versioning system if needed. We will continue to build on the React application so that for the next demo, we can demonstrate the basic functionality of the content management system.

Milestone 2 Acceptance Tests

For our milestone 2 demo, we showed our progress on the React application. We created an admin page where an admin could view a list of domains and then open the site associated with the domain to edit it. The site had a basic home page in which an admin could edit the title and description. The changes would be saved to the file server. We also created a sites page where the customers could go to view the pages, but could not edit them. The only thing that our client suggested was adding a way to create a domain.

Milestone 3 Acceptance Tests

For milestone 3 we needed to create 3 templates of our choice and include one integration in either the contact form or login template. We also decided to add functionality to create a new domain and choose which pages they wanted as part of their site. We were able to complete 2 of the templates and started 2 other ones. We underestimated the amount of time it would take to implement some of the functionality like image uploads and allowing the admin to add content, so some of the templates were not completed in time.

Milestone 4 Acceptance Tests

For milestone 4 we need to complete the rest of the templates and create 2 themes. We completed the blog, showcase, and gallery templates, and created the light and dark themes. One issue that we had run into was that implementing the image uploading was more difficult than we expected, so we just used default images for the templates that already had images and used an html form for image uploading. Our client approved the templates and themes, and said that we should implement image uploading for our milestone 5 demo.

Milestone 5 Acceptance Tests

For milestone 5 we finished image uploading, created sample content for a few pages, added an additional theme, and added Google Maps integration to our contact template. We were able to complete the image uploading and images could now be uploaded using inline editing. We also used inline editing and inline blocks to create the sample content for the pages. The blocks could be used to create reusable page content that can be edited to fit the page. We also added the minimal theme and gave users the ability to set the default location on an embedded Google Maps on the contact page. Our client suggested changing the user input from longitude and latitude for Google Maps component to an address.

Milestone 6 Acceptance Tests

For milestone 6 we completed the remaining features for the project. We created a template for BT reviews, completed the sample content for the remaining pages, added the ability to revert to previous versions of the site, and implemented SEO optimization. Our client suggested reworking the version reversion since we had it so they could choose to revert by date.

5 Implementation

Since we have already begun implementation during the first semester and were provided the implementation plan by our client, the design plan was created after we had already started the implementation. As a result, we followed most of our design plan. For this section we will discuss the specifics of the implementation.

5.1 FILE SERVER

The backend application for saving the site data is a Node Express rest API. This component is relatively simple and currently just consists of get and post endpoints for retrieving a list of sites, a list of domains, the site data, and for saving the edited site data. All json files representing the sites are saved to the file system of the machine on which the server is run on. When the files are saved

to the file server, they are named by the date and time they are saved, and whichever file was modified most recently is saved as “latest”. This file server will be replaced once we hand the project off to Buildertrend.

Endpoints:

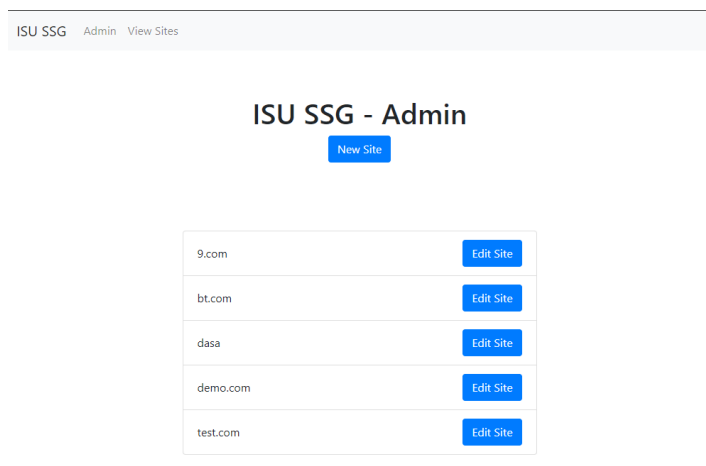
- Save File: Takes JSON data for a site and saves it to a file. If a previous version of the file already exists, it saves it as the latest version while keeping the previous version.
- Get File: Retrieves the latest file of a given domain
- Save Media: Takes a file and saves it to a given gallery for a given domain. Used for saving images that are not included in the site files.
- Get Gallery Images: retrieves the images from a given gallery.
- Domain List: Returns a list of created domains.
- Blog List: Returns a list of blogs from a given domain.
- Rollback: Sets the latest version of a site to a file that is closest to a given date.

5.2 CMS SITE

The application for creating and editing sites is a React with TypeScript web application that will use TinaCMS to create the content management system. We decided to use Bootstrap 4 for our user interface components and styling. We have already created an admin page which users can use to create and edit sites. The site also provides the user’s customers to view the latest and previous versions of the site. We created the home and contact templates for the sites, and have started the about us, services, and login pages. We will continue to build out the remaining pages and templates and will add features like themes, document uploads, and additional integrations.

Admin Page

The admin page consists of a list of domains that the user can edit. They can also create a new site from this page. Clicking on a domain will open the most recent version of the site.



New Site

This page allows an admin to create a new site. From this page, they can choose the site name, domain, theme, accent color, and the pages on the site.

ISU SSG - New Site


Company Name *

Domain *

Color Scheme *

Select ▼

Accent Color *



Pages

- About
- Login
- Contact
- Photo Galleries
- Project Showcase

View Sites Page

This page consists of a list of sites that a user can view, but cannot edit. It represents what a customer of the admin would see when visiting the site created by the admin.

ISU SSG - Sites

| | |
|--------------------------|---------------------------|
| 9.com/1618949218855.json | View Site |
| 9.com/1618949357277.json | View Site |
| 9.com/1618949431862.json | View Site |
| 9.com/1618949479793.json | View Site |
| 9.com/1618949498744.json | View Site |
| 9.com/1618949522508.json | View Site |
| 9.com/1618949531369.json | View Site |
| 9.com/1618949843994.json | View Site |
| 9.com/1618949853200.json | View Site |

Settings

The settings modal allows an admin to edit the company name, domain, theme, accent color, choose site pages, and set the default page.


Edit Site Close Save & Close

Company

Company Name *

Domain *

Color Scheme *

Accent Color *


Pages

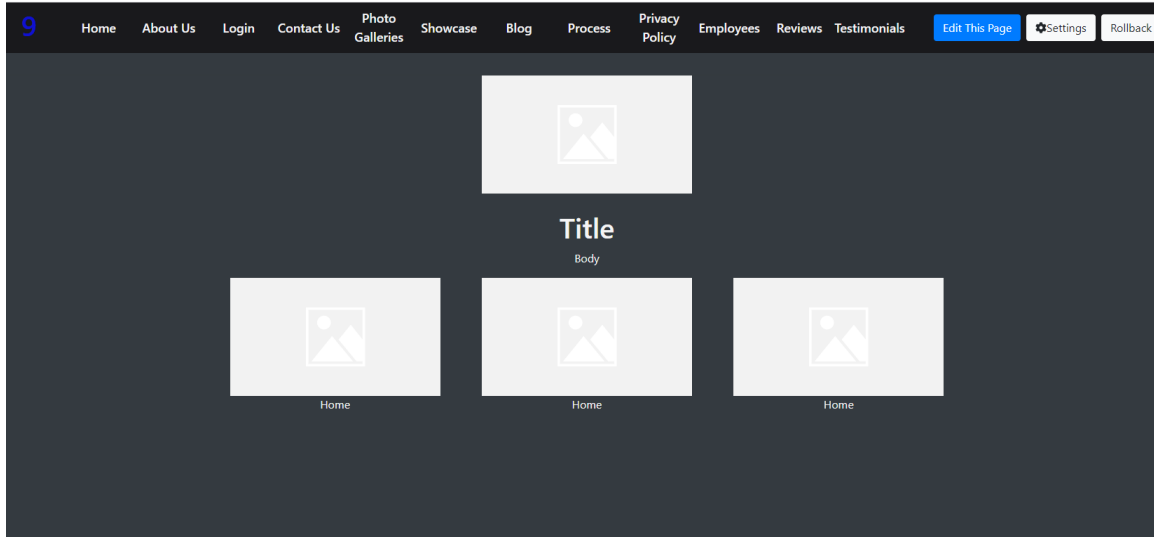
home Default

about Set as Default Page

Site Pages and Templates

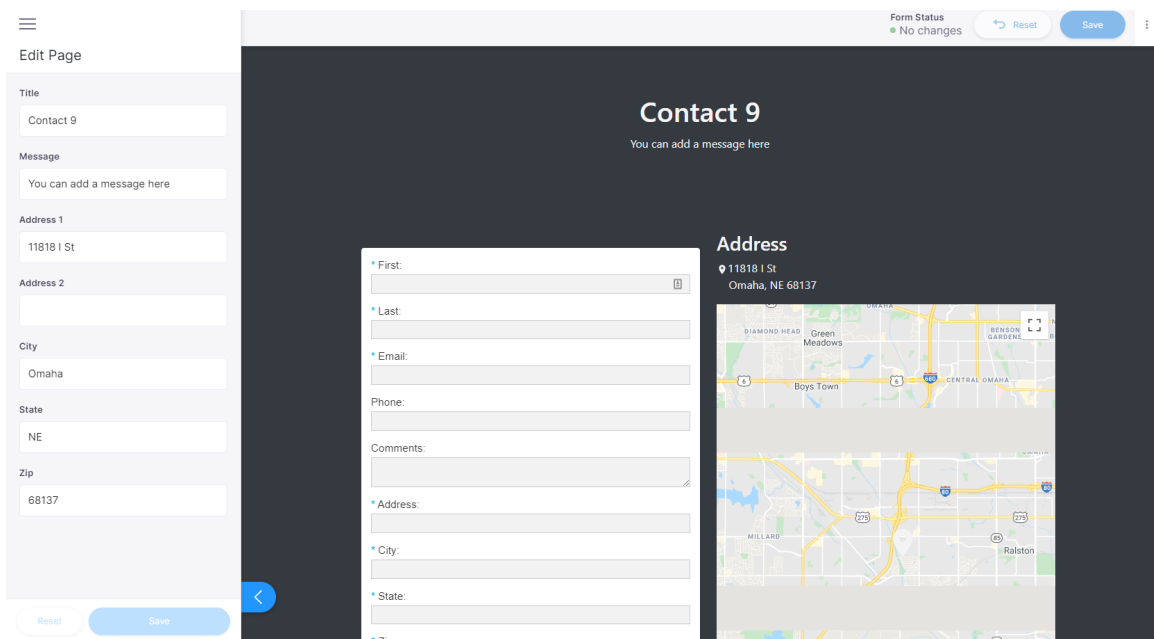
The pages and templates that can be added to a site are implementations of a base template. The templates contain content that cannot be removed such as embedded iframes, Google Maps integration, and videos. The rest of the pages are prepopulated with inline blocks to match the

mockups provided by Buildertrend. Admins can click the “Edit This Page” button to edit the content on the sites. If opened from the view sites page, the “Edit This Page” button will not be present.



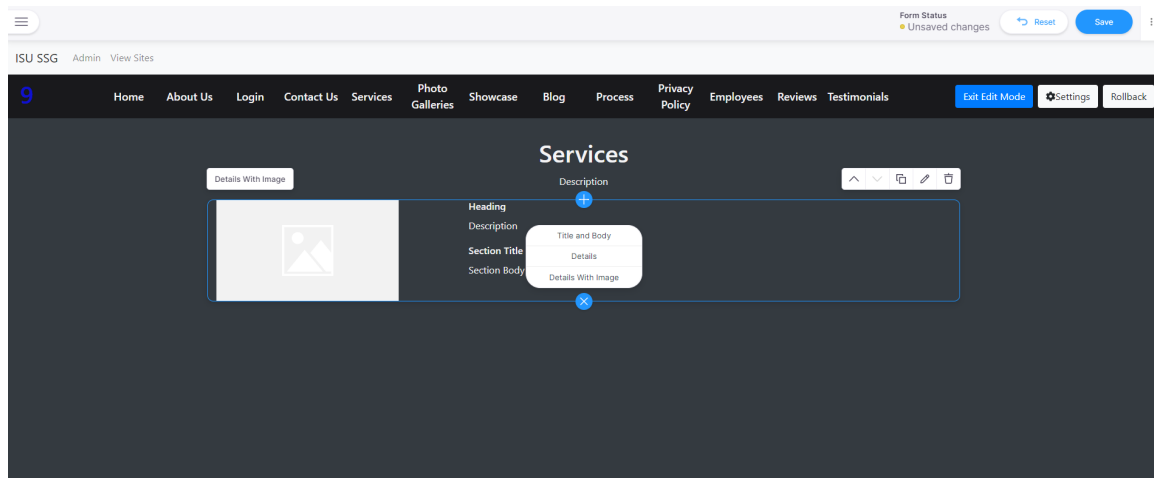
Editing Templates

Templates (Contact, Login, BT Reviews, Galleries, Blog, Showcase) can be edited from a sidebar form provided by TinaCMS. We set up the forms in the component for the template, and clicking save will save the changes to the backend and update the site.



Editing Pages with Sample Content

The remaining pages (Home, Services, Process, Privacy Policy, Employees, Testimonials, About) allow more freedom when it comes to choosing the sample content. These pages do not use the sidebar forms and instead use inline editing. The admin can choose blocks to add content to a page.



6 Closing Material

6.1 CONCLUSION

Over the course of the last 2 semesters, we created a website that Buildertrend will be able to use as a starting point for creating a CMS site for their customers. The website was built using React with TypeScript and TinaCMS and allows customers to create static sites using provided templates and pages. Buildertrend customers will be able to edit the content on the templates and pages to fit their companies. The sites will contain integrations with Buildertrend to connect customers with Buildertrend. The site content will be saved to a file server that will be replaced once we hand the project off to Buildertrend.

Since we were already given the implementation plan during our first semester and were expected to begin implementation at the time, we did not spend much time looking into other possible solutions. Buildertrend had already created a sample application with React with TypeScript and TinaCMS and had determined that this solution met the requirements for this project. The only component we had some flexibility with choosing our solution was the file server backend. However, Buildertrend recommended using Node, and we had already implemented the backend server with Node by the time we were supposed to begin looking at potential solutions.

Our solution will provide Buildertrend customers with a website design experience that should be tailored to their business type. The website's specific features should make website design quick and easy for business owners who do not have the skills or resources to create websites for their customers. This website should help provide Buildertrend customers with another reason why they should continue working with Buildertrend and hopefully attract new customers as well.

6.2 REFERENCES

- [1] Craig, D., 2015. *5 Reasons Not To Use Wix For Your Website – A Brief Wix Review*. [online] Fully Charged Media. Available at: <<https://www.fullychargedmedia.com/online-presence-fundamentals/5-reasons-not-to-use-wix-for-your-website-a-brief-wix-review/>> [Accessed 23 October 2020].
- [2] Garcia, J., 2020. *Wix Vs Squarespace – Squaring Up For A Good Fight*. [online] WebsiteToolTester. Available at: <<https://www.websiteooltester.com/en/blog/wix-vs-squarespace/>> [Accessed 23 October 2020].
- [3] Tovey, H., 2018. *6 Reasons Why You Shouldn't Use Squarespace - Heather Tovey*. [online] Heather Tovey. Available at: <<https://heathertovey.com/blog/why-you-shouldnt-use-squarespace/>> [Accessed 23 October 2020].

Appendix I: Operational Manual

A1.1 SETUP

Since we were only working on our local machines, the only way to access this project is to clone the repository. Since the repository is owned by Buildertrend, it is not a public repository and cannot be accessed without completing an Non-Disclosure Agreement. However if a user had access to the repository they would need to complete the following steps to setup the project:

1. Navigate to *isu-site-editor\back-end*
2. Run “npm run setup” or “npm i”
3. Navigate to *isu-site-editor\front-end*
4. Run “npm run setup” or “npm i --legacy-peer-deps”

A1.2 DEMO

After both the file server and React application have been set up, then both projects can be started.

1. Navigate to *isu-site-editor\back-end*
2. Run “npm run dev” or “nodemon”
3. Navigate to *isu-site-editor\front-end*
4. Run “npm run start” or “react-scripts start”

After the start scripts are run, the front end application can be accessed at localhost:3000. A user would then be able to create sites from the admin page and view those sites from the view sites page. All files would be saved to the user’s local machine. There is currently no way to deploy the sites or make them publicly available.

A1.3 TESTING

As previously mentioned, we did not write many tests for either the file server or front end application. Manual testing is the only way to test most of the functionality of the front end

application. This could be done by creating a site, editing the templates and pages, and changing the site settings. If a user wanted to run the few unit tests we did add they could do the following:

1. Navigate to *isu-site-editor\back-end*
2. Run “npm run test” or “jest”
3. Navigate to *isu-site-editor\front-end*
4. Run “npm run test” or “react-scripts test”

These tests have not been updated since milestone 1, so they most likely will not pass.