INDEx Full Stack App

Final Report

Sponsored by:

INDEx

JH Team

Members:

Josh Farr

Huy Tran

TABLE OF CONTENTS

I. Introduction 4

II. Team Members & Bios 5

III. Project Requirements Specification 5

III.1. Project Stakeholders 5

III.2. Use Cases 6

III.3. Functional Requirements 6

IV. Software Design 8

IV.1. Architecture Design 8

IV.1.1. Overview 8

IV.1.2. Subsystem Decomposition 9

IV.2. Data design 12

IV.3. User Interface Design 12

V. Test Case Specifications and Results 14

V.1. Test Objectives and Schedule 14

V1.1. Testing strategy 14

V.1.2. Test plans 15

V.2. Environment Requirements 16

V.3. Test Results 16

VI. Projects and Tools used 16

VII. Description of Final Prototype 16

VII.1. AWS API 16

VII.1.1 Functions and Interfaces Implemented 17

VII.1.2 Preliminary Tests 17

VII.2. AWS S3 17

VII.2.1 Functions and Interfaces Implemented 17

VII.2.2 Preliminary Tests 17

VII.3. Editor App Events tab 17

VII.3.1 Functions and Interfaces Implemented GET Dataset, PUT dataset, PUT Image 17

VII.3.2 Preliminary Tests 17

VII.4. Website Navigation bar 18

VII.4.1 Functions and Interfaces Implemented 18

VII.4.2 Preliminary Tests 18

VII.5. Website Homepage 18

VII.5.1 Functions and Interfaces Implemented 18

VII.5.2 Preliminary Tests 18

VII.6. Website Events Page 18

VIII.6.1 Functions and Interfaces Implemented 18

VIII.6.2 Preliminary Tests 18

VIII. Product Delivery Status 18

IX. Conclusions and Future Work 18

X. Acknowledgements 19

XI. Glossary 19

XII. References 19

XIII. Appendix A – Team Information 20

XIV. Appendix B - Example Testing Strategy Reporting 20

XV. Appendix C - Project Management 20

I. Introduction

INDEx, is an organization that guides the Independent Living Movement by advocating for choice, equity, and justice. There are many ways to develop a website, INDEx requires a solution that can be easily maintained and updated by their small team which only has a little experience with StudioPress Genesis. This document will provide all the information on INDex's new website, including its background, requirements, specification, and stages of work, and status.

1. Background and Related Work

INDEx [1], is an organization with people who work together to raise the voices of the disability community including those most marginalized – BIPOC, LGBTQIA2S+, immigrants, refugees, and those experiencing poverty. INDEx currently only has a small reference to their organization on their parent companies website. INDEx currently doesn't have a website and they are sharing the same website with their parent company DACNW.

WordPress is a content management system (CMS) to build and publish websites, it is the most popular CMS among others with the portion of 65.2% [3]. Genesis is the framework of Wordpress website, in other words, it decides how your website looks like. If WordPress is the engine of a car, then Genesis is the frame and body of it [2]. DACNW's website was built on WordPress and INDEx is expecting their website to be just as modular and user-friendly.

2. Project Overview

INDEx currently only has a small reference to their organization on their parent companies website (dacnw.org). INDEx plans on extending this reference to an entirely new website hosted under their own domain name. This means developing a website and making minor edits to the existing reference. The problem is that there are many ways to develop a website and INDEx requires a solution that can be easily maintained and updated by their small team with little web development experience.

As a nonprofit staffed by volunteers, INDEx doesn't have the means of hiring/supporting a web developer. Under these circumstances, the website is going to need to be developed in such a fashion that non-tech savvy individuals will be able to make edits/updates to the website when needed. With this objective, it is important to note that some of the staff members have minor experience using the Studiopress Genesis framework. Teaching/training INDEx employees on other web development techniques would most likely be unreasonable given the time constraint and potentially force them to outsource web developers after the project is complete. As stated previously, hiring web developers is not a desired outcome due to financial restrictions. Creating the webapp using the Studiopress Genesis framework would support all the desired outcomes with the least amount of risk. While this would be a suitable solution, it would make for a lackluster capstone project. With this in mind, we will have to develop a suitable alternative solution.

As stated previously, they only have minor experience with CMS frameworks. This means that training and documentation is still going to be necessary. Genesis blocks are a very intriguing option because of their templating and reusability. As an alternative, for example, we could create an events page that only requires the staff to fill out the information and not directly edit the build files. The documentation for this action would consist of using an application and a few steps on updating the page. This would allow us to focus more on the web app rather than detailed documentation on performing changes through code.

The final objective is to conform with the U.S Access Board's section 508 standards. Because INDEx is "an organization run by people with disabilities for people with disabilities", we have to ensure it is accessible to everyone. A specific example is avoiding serifs because these styles can be much more difficult to read if you have dyslexia. Thus, Sans-serif types should be used to accommodate this. A guideline that is often disregarded is the use of popup videos. This can be alarming to someone who is blind and difficult to turn off. We are going to have to follow these standards closely and work with the INDEx staff for any potential violations.

3. Client and Stakeholder Identification and Preferences

Our clients are INDEx's staff who will manage the website and its content (newsletter, events, blogs, etc). The primary stakeholders for this project will be the website's visitors. INDEx states "We work together to raise the voices of the disability community including those most marginalized". Clearly, the disability community and those marginalized are stakeholders in this project. The needs and preferences for both the client and stakeholders is for us to conform with the U.S Access Board's section 508 standards. Volunteers are another integral stakeholder in this project.

II. Team Members - Bios and Project Roles

Huy Tran is a computer science major who has knowledge and skills in designing and building websites using various programming languages and web development tools. Huy has a solid understanding of front-end development, including HTML, CSS, and JavaScript, as well as back-end development with languages such as PHP, Java and Python. Huy's expertise also extends to website architecture and optimization, including website security, performance, and accessibility. You are familiar with various website frameworks and content management systems such as Vuejs, and are able to customize them to meet specific client needs. For this project, Huy will be responsible for front-end-related objects including the website UI, and make sure it is functioning correctly.

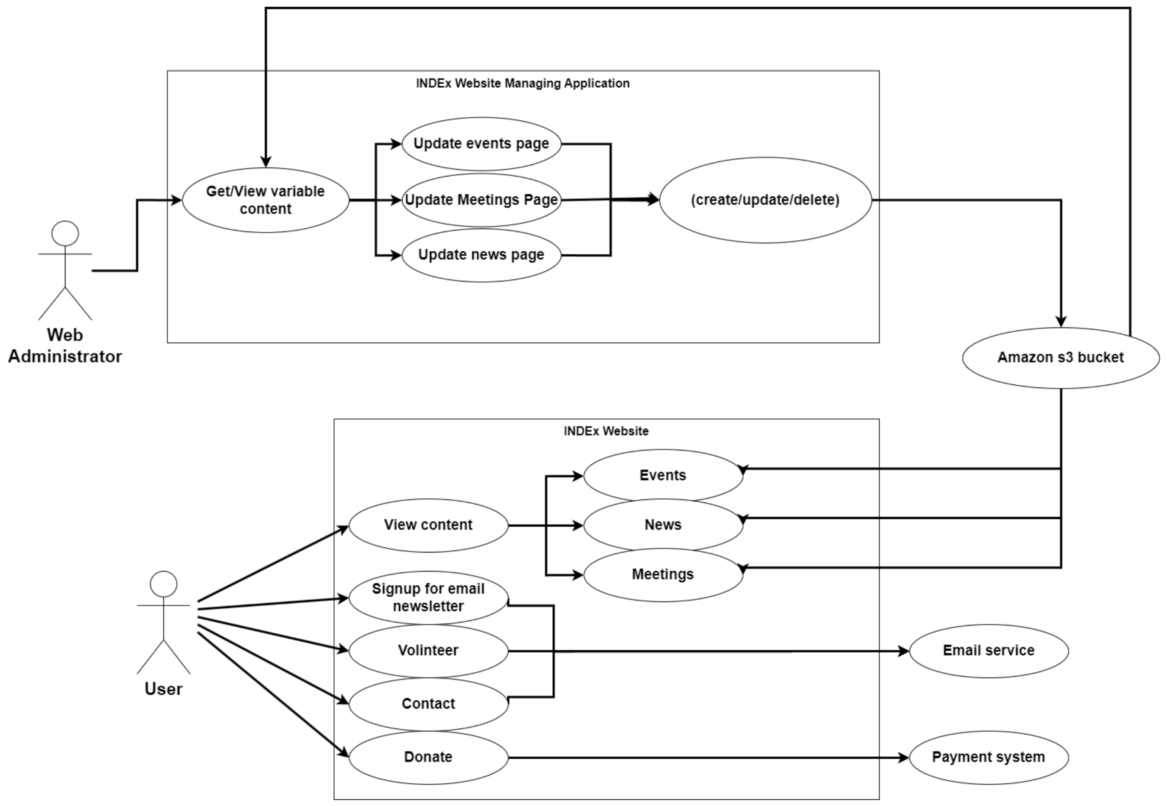
Josh Farr is a senior at Washington State University completing his bachelors in computer science. He has experience using web API's, cloud based platforms and creating software applications. Josh's focus is backend development and primarily uses languages such as C#, python and SQL. In this project, Josh is responsible for creating and maintaining AWS resources such as S3 and API Gateway. His other responsibility is developing the editor-app used to connect to the S3 and modify its json files which are dynamically read by the frontend to display its contents.

III. Project Requirements Specification

III.1. Project Stakeholders

The primary stakeholders for this project will be the website's visitors. INDEx states "We work together to raise the voices of the disability community including those most marginalized". Clearly, the disabled community and those marginalized are stakeholders in this project. The needs and preferences of both the client and stakeholders is for us to conform with the U.S Access Board's section 508 standards. Volunteers are another integral stakeholder in this project.

III.2. Use Cases



Story: Mels is responsible for updating/maintaining the INDEx website. She would like to add a new event to the events page so she opens the index-editor desktop app. After clicking the events tab, she decides to remove an old event so she clicks delete and removes it. She then fills out the required fields for a new event and clicks add.

Story: Thomas Bell is an advocate for the independent living movement. Thomas visits the INDEx website, learns about their mission and decides to volunteer. Thomas clicks the volunteer button on the homepage, fills out the register to volunteer form and submits it.

Story: Hannah Ellison is an advocate for accessibility. Hannah enjoys the newsletters on the index website and would like to be notified when they are posted. Hannah provides an email address to the newsletter signup box and clicks subscribe.

Story: Adam Parr likes to donate to missions he supports and visits the INDEx website. He clicks the donate button, fills out the billing and payment method information and confirms the payment.

Story: Eric Hughes would like to reach out to INDEx directly via email. He clicks the contact button, gives his name, email, message and clicks send.

III.3. Functional Requirements

III.3.1 (Events) INDEx editor application

Events page connected with a database: The “Events” page is functionally static, but the content is dynamically changing. The content of each event will follow a structured pattern and be stored in a Json file called “events”. When a user directs to the website, the json will be sent along with all of the other required build files.

connect to a database (a JSON file) where the latest as well as the older events are stored.
Source: The client originated this requirement.
Priority: Priority Level 3: Essential and required functionality

III.3.2 (Meetings) INDEx editor application

Meetings page connected with a database: Unlike the “Homepage” or “Contact us”, the “Events” page is not static since it always gets updated with new events from INDEx. Therefore, the page needs to connect to a database (a JSON file) where the latest as well as the older events are stored.
Source: The client originated this requirement.
Priority: Priority Level 3: Essential and required functionality

III.3.3 (News) INDEx editor application

News page connected with a database: Unlike the “Homepage” or “Contact us”, the “Events” page is not static since it always gets updated with new events from INDEx. Therefore, the page needs to connect to a database (a JSON file) where the latest as well as the older events are stored.
Source: The client originated this requirement.
Priority: Priority Level 3: Essential and required functionality

III.3.4 (Resources) INDEx editor application

News page connected with a database: Unlike the “Homepage” or “Contact us”, the “Events” page is not static since it always gets updated with new events from INDEx. Therefore, the page needs to connect to a database (a JSON file) where the latest as well as the older events are stored.
Source: The client originated this requirement.
Priority: Priority Level 3: Essential and required functionality

III.3.5 (Members) INDEx editor application

Members page connected with a database: Unlike the “Homepage” or “Contact us”, the “Events” page is not static since it always gets updated with new events from INDEx. Therefore, the page needs to connect to a database (a JSON file) where the latest as well as the older events are stored.
Source: The client originated this requirement.
Priority: Priority Level 3: Essential and required functionality

III.3.6 Mobile accessibility

Mobile compatibility: The website needs to be compatible with mobile devices (ex: smartphones and tablets)
Source: The client originated this requirement.
Priority: Priority Level 3: Essential and required functionality

III.3.7 Section 508 accessibility standards

Met Section 508 accessibility standards: The website needs to be compatible with mobile devices (ex: smartphones and tablets)
Source: The client originated this requirement.
Priority: Priority Level 4: Essential and required functionality

III.3.8 Email service

SMTP: the application needs to be able to connect INDEx with its user for contact, newsletters and volunteering. This will be done using a secure third party system.
Source: The client originated this requirement.
Priority: Priority Level 1: Desirable functionality

III.3.9 payment service

Donation system: the application needs to support donations through a secure third party API.
Source: The client originated this requirement.
Priority: Priority Level 0: Desirable functionality

III.4. Non-Functional Requirements

- **system shall be <maintainable>**

The website should require little maintenance and be easy to update. This is important because INDEx employees are not tech savvy.

- **system shall be <scalable>**

As a nonprofit organization, the ability to auto-scale is important because they want to keep costs low and only pay for resources when needed.

- **system shall be <reliable>**

the system should not crash. This is important because INDEx wouldn't know how to fix it.

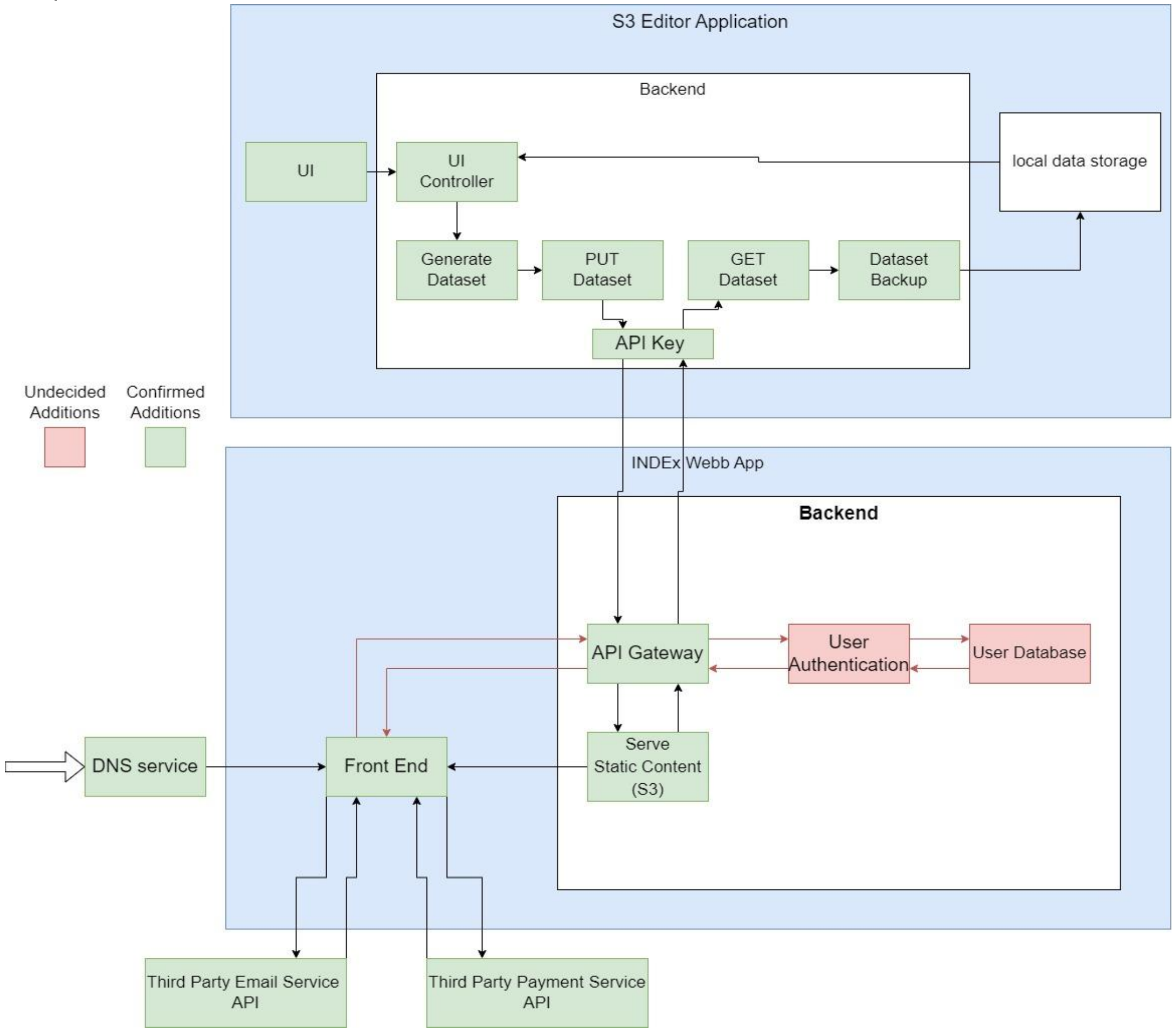
IV. Software Design

IV.1. Architecture Design

IV.1.1. Overview

INDEx needs a website but the staff are not technologically informed enough to make architectural decisions. This means we had to design it ourselves based on the requirements and our own decision making. An important factor in our design was ensuring its adaptability, there are still functionalities that have not been decided on so the architecture has to be modular and accommodating. The diagram below depicts the planned architecture and highlights what additions are decided(green) and undecided(red). The subsystem decomposition section will provide more detail on the individual components of the diagram. As shown, There are two main systems in our architecture, the web app and the S3 editor application. We decided to use a client side rendered architecture (commonly referred to as SPA's - single page applications) for a variety of reasons. Due to the nature of the INDEx website, a majority of the content is static making user experience and speed a priority. For public dynamic content, there is expected to be very little. For this reason, dynamic content is stored alongside the static content where the client side rendering will inject it upon building. This means having small datafiles as separate assets contained in the S3. Additionally, using the secure API, our editor application will be able to access and edit these data files directly. This will save on costs because there will be no server side computation necessary as well as no separate database. As an SPA framework, Vue.js will be used and suits this application well. For the editor application, a C# winforms application will be used. The choice to use amazon S3 static web hosting was due to its scalability and the ability to modify its contents through an API.

If they decide to support user functionality or other dynamic content, the API and front end can be modified to support the additions. For donations, email subscriptions and email contact, INDEx and our team will have to assess third party integrations. The chosen services will be implemented on the front end via API calls.



IV.1.2. Subsystem Decomposition

IV.1.2.1 API Gateway

a) Description

The API Gateway will serve as the connection between the editor application and the back end. It will be secured using an API key that will be embedded in the editor application. It will handle HTTP requests and will only allow the necessary operations to change the front end datasets. This means it will not allow access to any other S3 instance. It will also not allow the modification of any content besides the datasets. The datasets will also remain in JSON format to and from the S3 instance.

b) Concepts and Algorithms Generated

The API Gateway subsystem will ONLY allow HTTP GET and PUT methods. Given that this is an entrypoint to directly change the website, the API must follow the need-to-know principle of cyber security. That is, permissions to only what is necessary to accomplish the task.

c) Interface Description

Services Provided:

1. *Service name*: GetDataset{bucket}/{dataset} (GET)
   Service provided to: index editor app
   Description: Get{bucket}/{dataset} will read the corresponding JSON dataset from the S3 bucket containing the front end and return it back to the caller as JSON. Returns error to caller if: missing/incorrect API key, bucket or dataset parameters are incorrect or failure to get the dataset.

2. *Service name*: PutDataset{bucket}/{dataset} (PUT)
   Service provided to: index editor app
   Description: Put{bucket}/{dataset} will update the corresponding JSON dataset in S3 bucket containing the front end. Returns error to caller if: missing/incorrect API key, bucket or dataset parameters are incorrect or failure put the dataset.

3. *Service name*: PutImage (PUT)
   Service provided to: index editor app
   Description: PutImage will upload the corresponding image from local data storage to the S3 bucket containing the front end. Returns error to caller if: missing/incorrect API key, bucket or dataset parameters are incorrect or failure put the dataset.

Services Required:

1. *Service name*: INDEx(AWS_S3)
   Service provided from: INDEx(AWS_S3)

IV.1.2.2 INDEx(AWS_S3)

a) Description

The subsystem is responsible for hosting the web app.

b) Concepts and Algorithms Generated

The INDEx(AWS_S3) subsystem will be responsible for the distribution of the front end static content. It will contain the INDEx Vue.js app, that is, the CSS, Javascript, html and datasets such as events, meetings and news.

c) Interface Description

Services Provided:

1. *Service name*: INDEx(AWS_S3)
   Service provided to: AWS Route 53
   Description: S3 provides an entry point for Route 53 DNS service to redirect to

Services Required:

1. *Service name*: Domain Name
   Service provided from: AWS Route 53

IV.1.2.3 GET Dataset and PUT dataset

a) Description
Establishes the connection between the API Gateway subsystem and the INDEx editor app.

b) Concepts and Algorithms Generated
The GET/PUT dataset subsystems will be responsible for the retrieval and sending of datasets to and from the API Gateway. Upon opening the editor app, the get request will automatically be sent. Any subsequent PUT requests will be followed by a get request to verify success and maintain an active backup file.

c) Interface Description
Services Provided:

1. *Service name*: GET Dataset
   Service provided to: Dataset Backup,
   Description: After every GET request, send the dataset to Dataset Backup subsystem.

Services Provided:

2. *Service name*: PUT Dataset
   Service provided to: UI Controller
   Description: The service required for the INDEx editor app to send the updated dataset to s3

Services Required:

1. *Service name*: DELETE Dataset
   Service provided from: UI Controller
   Description: After every DELETE request, an event within the database is removed.

IV.1.2.4 Generate Dataset

a) Description
The logical step between the INDEx editor applications UI controller and the PUT subsystem.

b) Concepts and Algorithms Generated
Upon submission of dataset updates, the data must be converted into a JSON format. It is then added to the existing JSON and verified against a JSON schema. If successful, the dataset is sent to the PUT Dataset subsystem.

c) Interface Description
Services Provided:

1. *Service name*: Add {datatype}
   Service provided to: UI Controller
   Description: Processes user input into json format to be sent.

Services Provided:

1. *Service name*: PUT Status
   Service provided to: UI Controller
   Description: Event to notify the UI Controller of a successful or unsuccessful addition to dataset.

IV.1.2.5 UI controller

a) Description
Handles the current data, user input data and events to be displayed to the UI.

b) Concepts and Algorithms Generated
The UI controller fetches the most recent datasets from backup for the UI. It handles events such as button clicks and user input.

c) Interface Description
Services Provided:

1. *Service name*: Events
   Service provided to: UI
   Description: Notifies users of actions taken and status of commands made.

2. *Service name*: Input verification
   Service provided to: UI
   Description: Notifies users of invalid input.

Services Required:

1. *Service name*: Notification
   Service provided from: UI Controller

IV.1.2.6 Dataset Backup

a) Description
Retrieve from and store datasets into a backup file.

b) Concepts and Algorithms Generated
On every GET dataset request, this subsystem stores the retrieved dataset into a backup file. Each dataset will have a timestamp and will not persist longer than a determined time.

c) Interface Description
Services Provided:

1. *Service name*: Get backup
   Service provided to: UI controller
   Description: sends a backup dataset to the UI controller

2. *Service name*: Create Backup
   Service provided to: Automatic
   Description: Creates a backup of the retrieved dataset

Services Required:

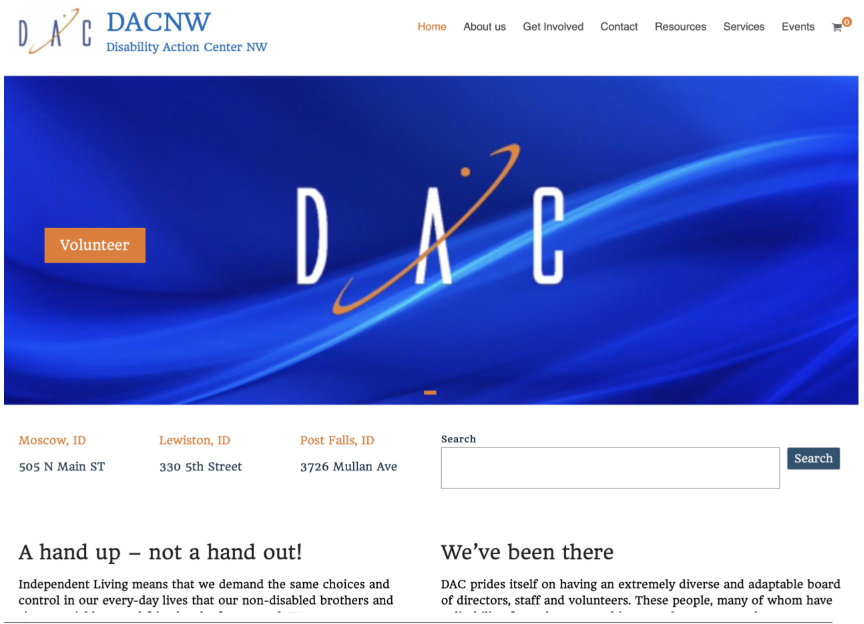
1. *Service name*: Backup
   Service provided from: UI Controller

IV.2. Data design
There are many data objects required for the success of this project. All data is in JSON format and each datafile corresponds to a “page” of the website. The contents in each JSON is a dynamic representation of a specific webpage which typically consists of text and image urls.

The “events.json”, “memebrs.json”, “news.json” and many more structures of these JSON objects can be found in the front-end section of the repository.

IV.3.1 Frontend User Interface Design

The website UI is similar to DACWN with a navigation bar to redirect to Home, About us, Get Involved, Contact, Resources, Services, and Events. Other than the navigation bar, the content for each page is currently empty, but will soon be implemented once we have further discussion with our clients. So far, what we have in mind is the pages would be visually like these images (provided by the clients):

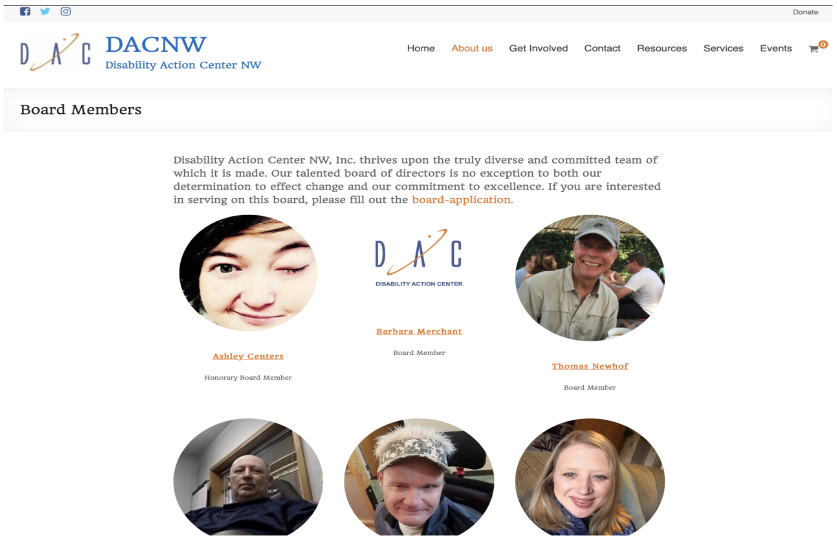


A hand up – not a hand out!

Independent Living means that we demand the same choices and control in our every-day lives that our non-disabled brothers and

We’ve been there

DAC prides itself on having an extremely diverse and adaptable board of directors, staff and volunteers. These people, many of whom have



Disability Action Center NW, Inc. thrives upon the truly diverse and committed team of which it is made. Our talented board of directors is no exception to both our determination to effect change and our commitment to excellence. If you are interested in serving on this board, please fill out the board-application.

IV.3.2 Backend User Interface Design

The UI for the editor-app application is stylistically simple but functionally complete. While more time could be spent improving the aesthetics of the editor-app, we believe that it is a tool and that functionality should be prioritized. Early into development, time was spent drawing UI concept art but was quickly abandoned for the reasons stated. This is not a public facing application and will only be used by a handful of index employees. For these reasons, the UI is simple with textboxes, buttons, tabs and dropdown menus. Keeping the UI simple improves the deployment speed of new features and provides more time to be spent optimizing and maintaining its key purpose, functionality.

V. Test Case Specifications and Results

V.1. Test Objectives and Schedule

We will be using testing to ensure the website works smoothly and error-free before handling it to the customer. There will need to be testing for two different components, the front-end and the back-end.

The front-end, which is the website, includes static pages (homepage, about us, etc) and some dynamic pages such as News and Events. The static pages only need to test to ensure they display appropriately on both desktop and mobile. Dynamic pages will have to work with the JSON file (where the events and news are stored) in addition to the static pages' requirement.

The back-end, which is the winform application, includes text boxes and buttons for the user to fill in and push updates to the JSON file. The application will need to test to make sure the events get to the JSON file located on the S3 server.

We will be using Agile Process for this project, applied for both front-end and back-end. Additionally, we will use Github Actions as our CI/CD tool to test the website. Github Actions syncs our Vue application with our AWS S3 bucket. This means that the S3 bucket will always contain the most recent successful build of the website. Any change pushed to the github repository that does not build successfully will not be deployed to S3..

Agile Methodology
Sprint 1
Sprint 2
Sprint 3
Plan
Design
Build
Test
Review
Launch

Our deliverables include our Github repository with the code and documentation for the product. Milestones in our testing process include writing, testing and integrating our code with CI/CD (Github action) as well as demos and user testing the code. These milestones may be achieved multiple times in our development lifecycle.

V1.1. Testing strategy

We will be testing our code from user and admin point of view to ensure the final product works smoothly and satisfies our Software Requirements Specification.

a. **Writing the code:** Both of us are responsible for different parts of the projects including Website front-end (UI), back-end (user database, login system), and Winform app.
b. **Testing the codes:** Each developer tests their code ensuring it's working and compatible with other related components before pushing them to Github. Broken/buggy code should never be pushed.
c. **Push to Github:** Commit and then push the tested code to Github.
d. **Merge to main branch:** Create a merge request and merge the developing branch to the master/main branch.
e. **Github CI/CD check for validation:** Github Actions will check for errors. If there is none, the merge request will succeed and the website is successfully deployed. Otherwise, the merge request will fail and the developer will go back to step a.

V.1.2. Test plans

V.1.2.1. Unit testing

Our team will conduct unit testing by testing each individual unit/feature/function before putting it into our code. This act is to ensure the correct functionality and compatibility with other functions. Any functions that accept input from a user should be thoroughly tested with invalid inputs. Functions writing directly to the events, news and meetings json data should be tested with utmost care.

V.1.2.2. Integration Testing

Considering we have already prepared the solution approach and use cases, testing integration will be achieved by testing the communication between functions for errors or mismatch. This will ensure the correct functionality/communication between all features/functions presented in the solution approach.

V.1.2.3. System Testing

V.1.2.3.1. Functional testing

This system testing plan will encompass the Index editing app pushing and retrieving data from the website. The functional requirements section includes the news, events, and meetings pages. Each page will be tested using the same method. This will test the functionality of the editor app retrieving/sending data, the API passing data, the s3 storing data, and the website's ability to render that data. First, we will generate many JSON documents with a mixture of valid, invalid or missing data. Pushing these documents using the app should not result in any crash and instead should either fail to render or be recognized as invalid/corrupt data. Our API and s3 logs will be monitored throughout for errors.

V.1.2.3.2. Performance testing

Similar to the functional system test, the editor app will be used to push and retrieve data. The first system performance test will be with large data sets. The editor app will push increasingly large datasets until it fails or far surpasses requirements. This test will also show the effects it has on the website, S3 and API. The website should maintain a reasonable load time and the API and S3 logs will be checked for errors.

The second test will require creating a simple testing tool. This tool will make many small push requests to send data to the website. The data will be sent at the maximum rate limit of the API. AWS logs will be monitored live and the website will be referenced for accuracy. While it is very unlikely the editor app will ever send updates at the rate limit, some errors may arise that aren't necessarily due to the rate.

V.1.2.3.3. User Acceptance Testing

The project will be tested several times, from the user perspective, throughout the building and testing processes. Additionally, the clients are the ones who will set the requirements of acceptance, therefore the system will be tested until it reaches the client/user acceptance.

V.2. Environment Requirements

The testing environment will make full use of AWS metrics and logs. The S3 and API Gateway services we are using report important metrics in regard to performance and also general errors. We do not expect these fully managed services to have problems but if they do we can implement further logging using CloudWatch Alarms or CloudTrail.

GitHub Actions will be used for running the CI/CD of the front-end website. On push, the runner attempts to build the website and reports either a failure or success. On success, the website will be deployed to AWS.

In order to unit test the editor application, NUnit will be used. Both NUnit and WinForms are part of the .NET framework making for good compatibility.

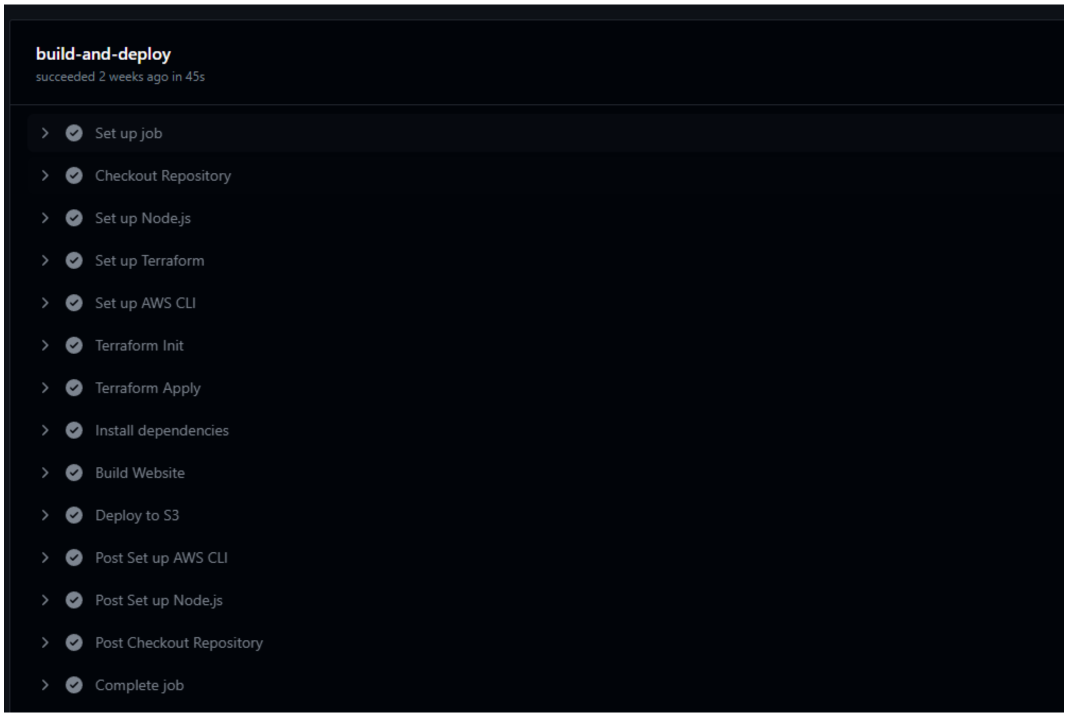
V.3. Test Results

Front-end (website):

Most of the features on the website are tested manually by clicking through each page, button, and link to ensure they are working as expected. Mobile compatibility is tested by adjusting the screen to different resolutions to ensure everything displays flawlessly on both desktop and mobile. Data that are saved to display (events, news, etc) are also tested multiple times, verifying the import functions work correctly and there is no misdisplay. Github action also checks for errors before pushing the new update to the S3 bucket, making the website Vuejs error-free.

Back-end (Vue app):

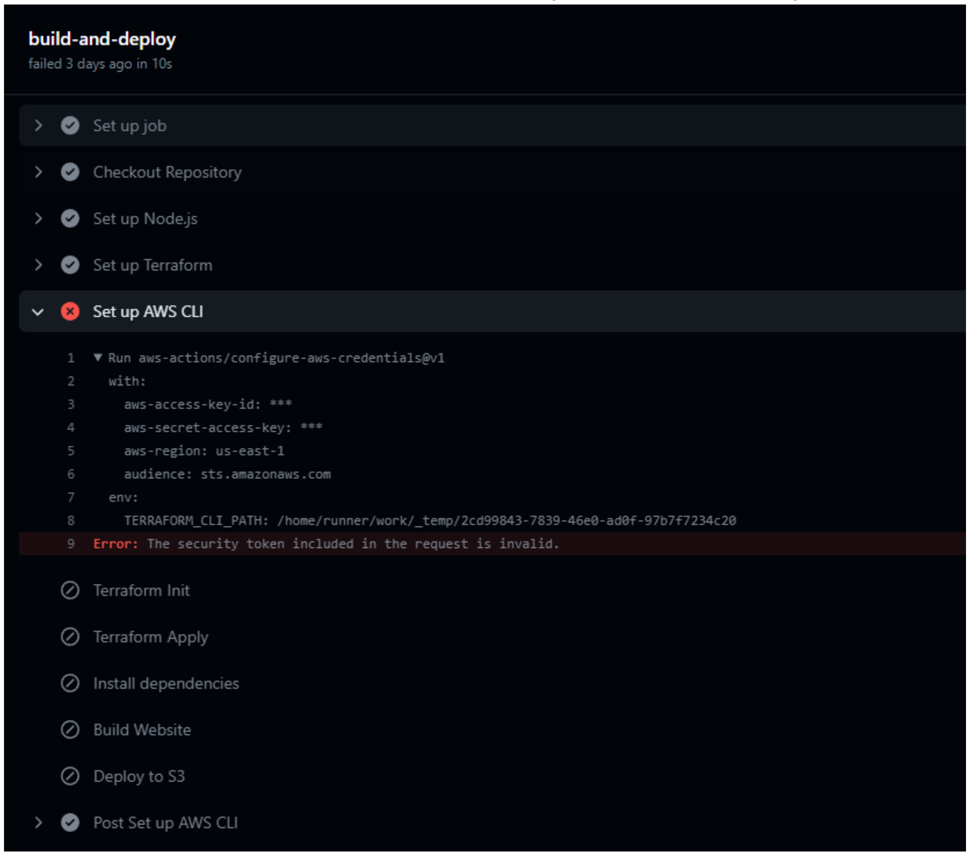
We use Github Action to test our work before merging it to our main branch and deploying the app. Test output for a successful deployment (expected result) is shown below:



```
build-and-deploy
succeeded 2 weeks ago in 45s

Set up job
Checkout Repository
Set up Node.js
Set up Terraform
Set up AWS CLI
Terraform Init
Terraform Apply
Install dependencies
Build Website
Deploy to S3
Post Set up AWS CLI
Post Set up Node.js
Post Checkout Repository
Complete job
```

If the test fails, causes of failure are also provided, an example is shown below:



```
build-and-deploy
failed 3 days ago in 10s

Set up job
Checkout Repository
Set up Node.js
Set up Terraform
Set up AWS CLI
▼ Run aws-actions/configure-aws-credentials@v1
  with:
    aws-access-key-id: ***
    aws-secret-access-key: ***
    aws-region: us-east-1
    audience: sts.amazonaws.com
  env:
    TERRAFORM_CLI_PATH: /home/runner/work/_temp/2cd99843-7839-46e0-ad0f-97b7f7234c20
Error: The security token included in the request is invalid.
Terraform Init
Terraform Apply
Install dependencies
Build Website
Deploy to S3
Post Set up AWS CLI
```

VI. Projects and Tools used

AWS S3 Bucket	A cloud-based object storage service provided by Amazon Web Services (AWS). An S3 bucket is a container for storing objects (files) in Amazon S3
AWS API Gateway	Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.
Visual Studio Code	Visual Studio Code is the development environment used to write and run the project locally.
Github	A web-based platform that provides version control, collaboration tools, and other features for software developers to host and review code.
Vue.js	A progressive JavaScript framework for building user interfaces that is lightweight, flexible, and easy to learn.
Winforms	A UI framework for building Windows desktop apps.

Languages Used in Project				
JavaScript	CSS	HTML	Terraform	C#

VII. Description of Final Prototype

The Final Prototype required standing-up infrastructures such as an AWS S3 bucket and API. The S3 bucket is configured for static web hosting and it contains all of the build files for the website and the raw JSON file used for the events page. Right now the editor app has the capability to make an HTTP get request for the events JSON, make modifications to it, and use an HTTP put call to store it back in the S3 bucket. The editor app can also create new events and upload images to the S3. This is fully functional except for one minor part that needs to be added. When uploading an image using the editor app, the app needs to store a link to that image in JSON. The website currently has a homepage and events page which are also accessible on mobile. The events page dynamically imports the events JSON and correctly displays it.

VII.1. AWS API

The AWS API Gateway is currently being used for the events and uploading of images. It is already configured to support a meetings page and a new page also. It makes use of the built-in API key generator to secure our endpoints. The AWS Identity and access management (IAM) has also been implemented to restrict the API to only be able to access the S3 bucket and the specified files within that S3 bucket.

VII.1.1 Functions and Interfaces Implemented

See appendix A image 1 for all currently implemented API endpoints. Briefly, they are for events, news, meetings, and images, all of which implement GET, PUT, and DELETE.

VII.1.2 Preliminary Tests

Using some basic end-to-end tests and postman we were able to verify the endpoints were working correctly. Using postman we would send a new JSON and then check to see if the S3 contents reflected the change. We also tried to access the endpoints without the API key and failed. Finally, with the API key, we attempted to access unspecified files such as the website build files, and got the current response that indicated the files were not accessible.

VII.2. AWS S3

The AWS S3 bucket is a static container for our website. Right now it is publicly available and servers the build files of the website. Because we are using Vue Js, an (SPA), we will be able to implement all the required functionality using S3. Setting up GitHub actions to deploy the website to S3 on push was also easily achieved.

VII.2.1 Functions and Interfaces Implemented

The S3 currently only has a few endpoints that are handled and maintained through AWS. These endpoints are with the API and the open endpoint serving the website.

VII.2.2 Preliminary Tests

Testing the S3 was a fairly quick process. At the beginning of the project, we created the S3 and CI/CD. Once those were in place we pushed the default Vue app and saw that the S3 contained the correct files and the endpoint retrieved the correct files.

VII.3. Editor App Events tab

The editor app currently has the events tab fully functional except for one minor addition that still needs to be worked out. Upon starting, the app tests the connection to the website, if the connection fails an error response is given and in the future, we will add documentation as to how it may be fixed. If the connection is successful, then the event's JSON was successfully retrieved from the S3 and they are displayed in the events tab. The user can create, edit and delete events and finally upload any modifications to the S3. There is also limited input verification such as missing input or minor grammatical errors. We plan on extending the validation process to ensure only correct information is uploaded.

VII.3.1 Functions and Interfaces Implemented GET Dataset, PUT dataset, GET Image, PUT Image

These are the functions implemented in the editor app, specifically, for the events tab. These functions will be extended for many other endpoints.

VII.3.2 Preliminary Tests

End-to-end testing has been performed to verify the correct retrieval and sending of the events JSON. Other tests exist within the app to verify the necessary data is added. N Unit tests

VII.4. Website Navigation bar

VII.4.1 Functions and Interfaces Implemented

The navigation bar was implemented on top of the website and it consists of 2 main things: website branding and navigation buttons. The branding part contains the Index logo and the company name. The navigation contains 7 buttons (subject to change) that redirect the visitor to different pages labeled on each button.

VII.4.2 Preliminary Tests

The navigation bar was tested manually by adjusting the screen to different resolutions, this makes sure it displays flawlessly on both desktop and mobile.

VII.5. Website Homepage

VII.5.1 Functions and Interfaces Implemented

A homepage banner is placed center and at the top of the homepage. Below the banner are informative texts and images provided by the clients. An event section, at the bottom of the homepage, shows the latest 3 events.

VII.5.2 Preliminary Tests

The homepage was tested manually by adjusting the screen to different resolutions, this makes sure it displays flawlessly on both desktop and mobile. For the events displayed at the bottom of the homepage, a function to export the events within the file events.json is added and Vue takes care of the exported events by looping through the events and displaying them under HTML.

VII.6. Website Events Page

VIII.6.1 Functions and Interfaces Implemented

Vue Router is implemented, during the event page’s addition, in order to redirect page to page smoothly. A function to export the events within the file events.json is added and Vue takes care of the exported events by looping through the events and displaying them under HTML.

VIII.6.2 Preliminary Tests

The homepage was tested manually to ensure the events are imported correctly from the events.json file. Mobile compatibility is also tested by adjusting the screen to different resolutions to see if there is anything displayed incorrectly.

VIII. Product Delivery Status

The deliverables of this project include:

- This report
- A readme/how to setup docs
- Instructions for the desktop's editor app
- Instructions for creating AWS resources
- Front end VUE code
- Editor app Winforms code
- Infrastructure Terraform code

The project is set to be delivered to the clients on April 30th and this document is to be finished before May 2nd. A meeting to guide the clients could also be set up upon the client's request. Leading up to the delivery, time will be spent polishing and updating setup instructions to allow further work to be done on the project. This will be especially important for less documented aspects such as AWS resources.

IX. Conclusions and Future Work

IX.1. Limitations and Recommendations

The website and editor app: One of the limits is that the client cannot change the layout of the website or move the elements around. In the current state of the application, the editor app does not provide any tools or methods to move elements nor has it been tried.

X.2. Future Work

The website could be extended by implementing more features such as donations, contact us page functionality and newsletter functionality. The editor app can be aesthetically improved, changing the layout to help the user experience become better. Any additional pages upon the client's request will need to be added to the website along with a tab in the editor app. Adding themes and the ability to change them could be greatly expanded. Themes pertaining to color, font, font size, and perhaps position.

X. Acknowledgements

We would like to thank Vicky and the people at INDEx for giving us an opportunity to work with them and to AJ for helping us throughout the semester. AJ's willingness to listen and accommodate helped us achieve the best possible outcome. Thank you, AJ for your guidance.

XI. Glossary

INDEx: an organization that guides the Independent Living Movement by advocating for choice, equity, and justice

WordPress: open-source content management system

StudioPress: themes and design framework for WordPress

Genesis: WordPress framework

Vue.js framework: A way to build a website.

S3: Formally called Simple Storage Service, S3 is an AWS service.

AWS: Amazon web services (AWS) is a cloud service provider.

API: Application Programming Interface (API) can be thought of as a contract of service between two applications.

UI: User Interface (UI) is anything a user may interact with to use a digital product or service

XII. References

1. "INDEx Inland Northwest Disability Experience" dacnw.org
https://dacnw.org/contact/index/ (accessed Sept. 28, 2022)

2. "Frequently Asked Questions" studiopress.com
https://www.studiopress.com/faqs/ (accessed Oct. 5, 2022)

3. A.Fitzgerald."20 WordPress Statistics You Should Know in 2022". Hubspot.com
https://blog.hubspot.com/website/wordpress-stats (accessed Oct. 5, 2022)

4. "TESTING IN AN AGILE PROJECT?". stardust-testing.com
https://www2.stardust-testing.com/en/blog-en/testing-in-an-agile-project
(accessed Nov. 7,2022)

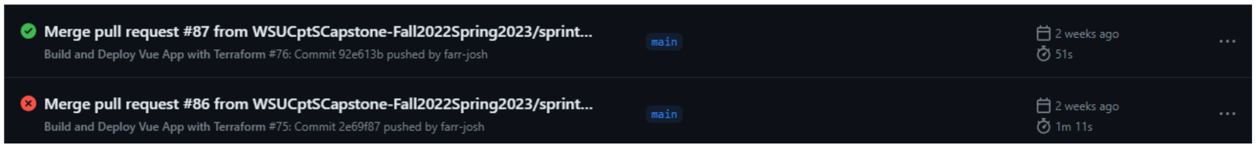
5. "C# documentation". https://learn.microsoft.com/en-us/dotnet/csharp/
(accessed Sep. 15,2022)

XIII. Appendix A – Team Information

Josh Farr - josh.farr@wsu.edu
Huy Tran - huy.tran1@wsu.edu

XIV. Appendix B - Example Testing Strategy Reporting

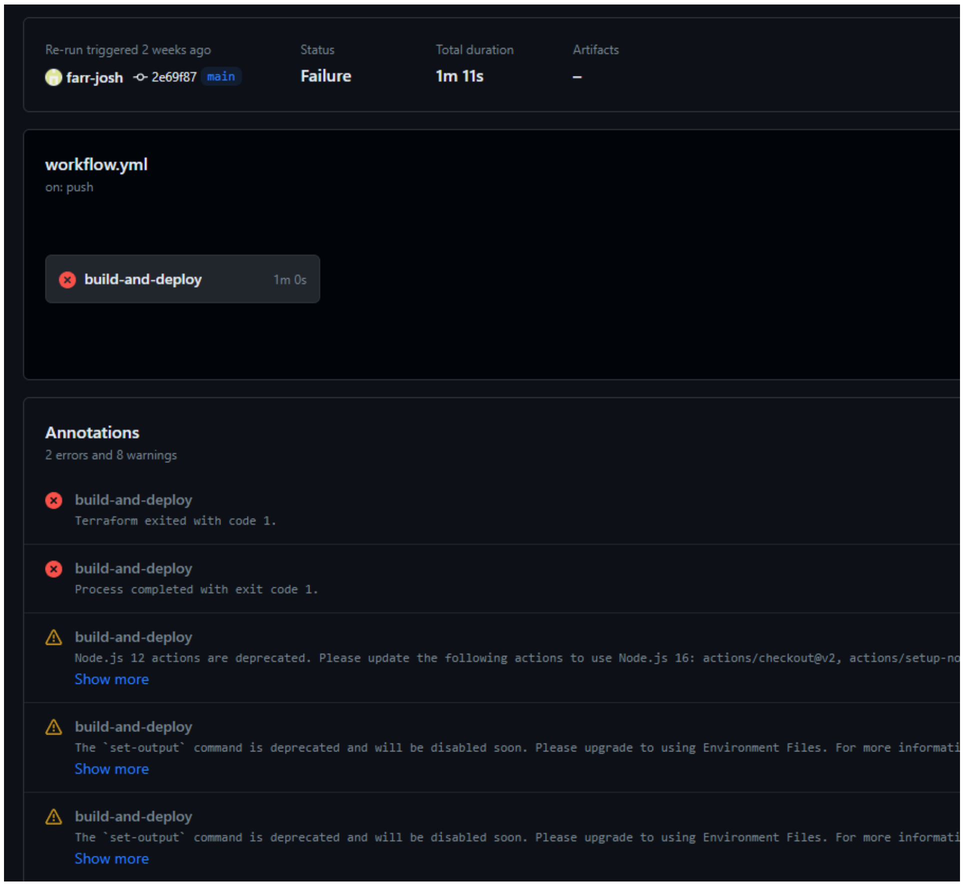
We build and deploy the Vue app through Terraform and use GitHub Action to test our work. If there is no issue with our code, the merge pull request will come through.



✓ Merge pull request #87 from WSUCptSCapstone-Fall2022Spring2023/sprint...
Build and Deploy Vue App with Terraform #76: Commit 92e613b pushed by farr-josh
main
2 weeks ago
51s

✗ Merge pull request #86 from WSUCptSCapstone-Fall2022Spring2023/sprint...
Build and Deploy Vue App with Terraform #75: Commit 2e69f87 pushed by farr-josh
main
2 weeks ago
1m 11s

If there is an issue with our code and it prevents the Vue app from deploying, we can see the details which include the cause of failure and warnings on GitHub Action.



Re-run triggered 2 weeks ago · farr-josh · 2e69f87 · main

Status: Failure

Total duration: 1m 11s

Artifacts: –

workflow.yml
on: push

build-and-deploy 1m 0s

Annotations
2 errors and 8 warnings

build-and-deploy
```
Terraform exited with code 1.
```

build-and-deploy
```
Process completed with exit code 1.
```

build-and-deploy
```
Node.js 12 actions are deprecated. Please update the following actions to use Node.js 16: actions/checkout@v2, actions/setup-no
```
Show more

build-and-deploy
```
The `set-output` command is deprecated and will be disabled soon. Please upgrade to using Environment Files. For more informati
```
Show more

build-and-deploy
```
The `set-output` command is deprecated and will be disabled soon. Please upgrade to using Environment Files. For more informati
```
Show more

Example link of failed deployment:
https://github.com/WSUCptSCapstone-Fall2022Spring2023/index-fullstackapp/actions/runs/4827704856
Example link of successful deployment::
https://github.com/WSUCptSCapstone-Fall2022Spring2023/index-fullstackapp/actions/runs/4749123461

XV. Appendix C - Project Management

Our team communicates mostly through Discord and we host a meeting with our client through Zoom every 2 weeks. The typical client meeting normally goes in this order:

1. Report on current progress
2. Demo what we have done or been working on the past 2 weeks
3. Discussion of issues and improvement
4. Planning future work for the meeting.

This model works well for us as it let us show our work as well as keeps the client up to date.