INDEx Full Stack App

Sponsored by:

INDEx

JH Team

Members:

Josh Farr

Huy Tran

TABLE OF CONTENTS

I. Introduction

INDEx, is an organization which 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 website, 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 use the same model.

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. Creating the

webapp using the Studiopress Genesis framework will support all the desired outcomes with the least amount of risk. An added benefit to this framework is that dacnw, the parent company, is already using Studiopress Genesis and can potentially provide support to INDEx if needed.

As stated previously, they only have minor experience with the framework. This means that training and documentation is still going to be necessary. Genesis blocks are a very intriguing option because of the templating and reusability. For example, we could create an event block that only requires the user to drag and drop it onto the page and fill out the information. The documentation for this action would consist of the location and name of the block and a few steps on attaching it. This would allow us to focus more on the webapp rather than documentation. Another use case for Genesis is that custom page templates can be prepared for future implementations. For example, they have a surplus of volunteers right now and don't want a volunteer signup page. In the future, they might need more volunteers and a signup page. To do this they would simply click “add page” and select our template that was pre-prepared to their liking.

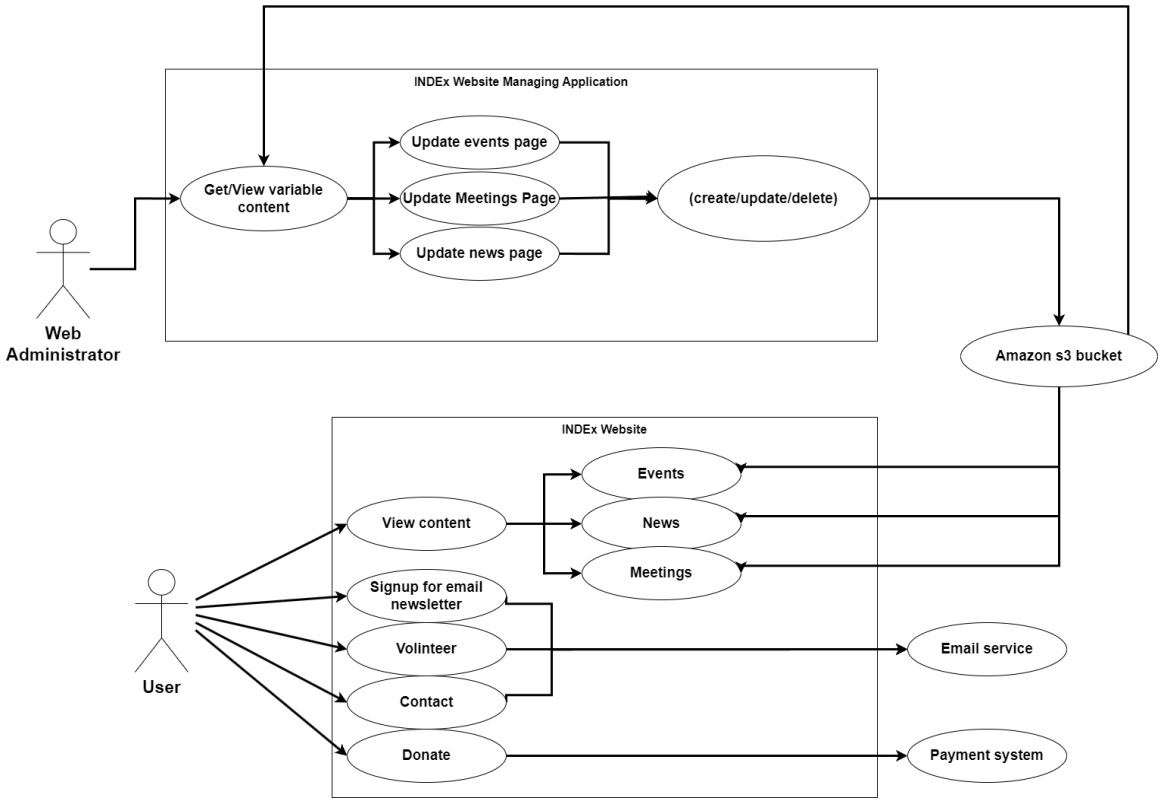
The final objective is to conform with the U.S Access Board’s section 508 standards. Because INDEx is “an organization run by people with disabilities for people with disabilities”, we have to ensure it is accessible to everyone. A specific example is avoiding serifs because these styles can be much 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. Project Requirements

II.1. 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: HannahEllison 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.

II.2. Functional Requirements

II.2.1 INDEx editor application

The website would have pages where administrators and staffs add and edit the contents such as news or events.The winforms application is a program to help them easily do such things.
Source: The team originated this requirement. It is necessary for the admin/staff to update the contents on the website.
Priority: Priority Level 3: Essential and required functionality

II.2.2 (Events) INDEx editor application
Events 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: Desirable functionality

II.2.3 (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

II.2.4 (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

II.2.5 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

II.2.6 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

II.2.4.1 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

II.2.4.2 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

II.3. 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.

III. System Evolution

The website is hosted on an Amazon AWS S3 instance and it is based on Vue.js framework.The website contains a navigation bar which is located at the top of all pages, visitors can use the navigation bar to go to homepage, events, newsletter and blogs. For events, newsletter and blogs, there will be a database to store all of those, they can be edited through a winforms application running on a desktop. The risk we might run into is someone deobfuscated the winforms application and freely add/delete the website's events.In addition, since the website has static pages, it would be difficult to change things around if the clients wish to.

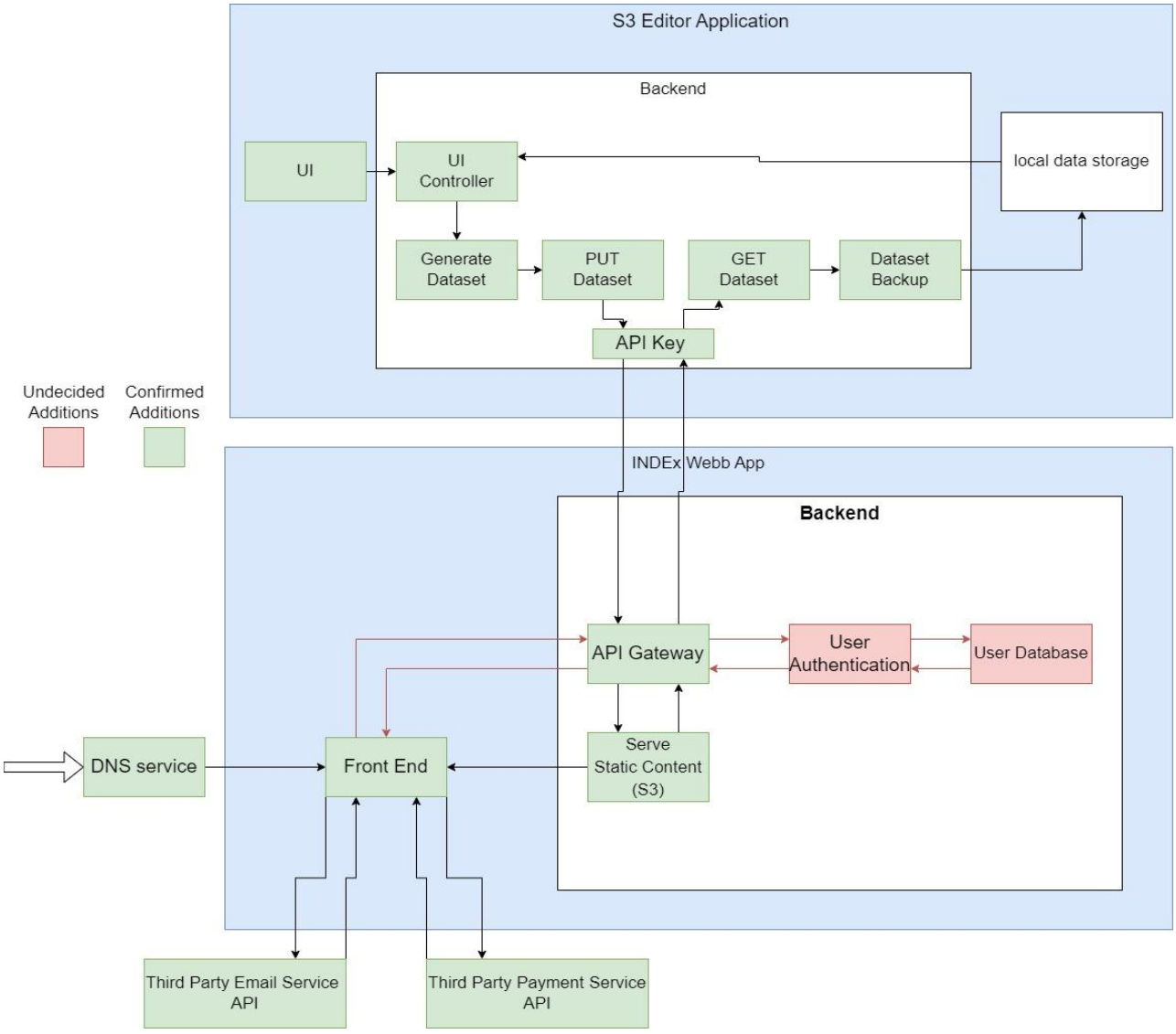
IV.Solution Approach

IV.1. Architecture Design

IV.1.1Overview

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.2. Subsystem Decomposition

IV.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: Get{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.

Services Required:

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

IV.2.2 INDEx(AWS_S3)

a) Description

The subsystem 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.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.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.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.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:

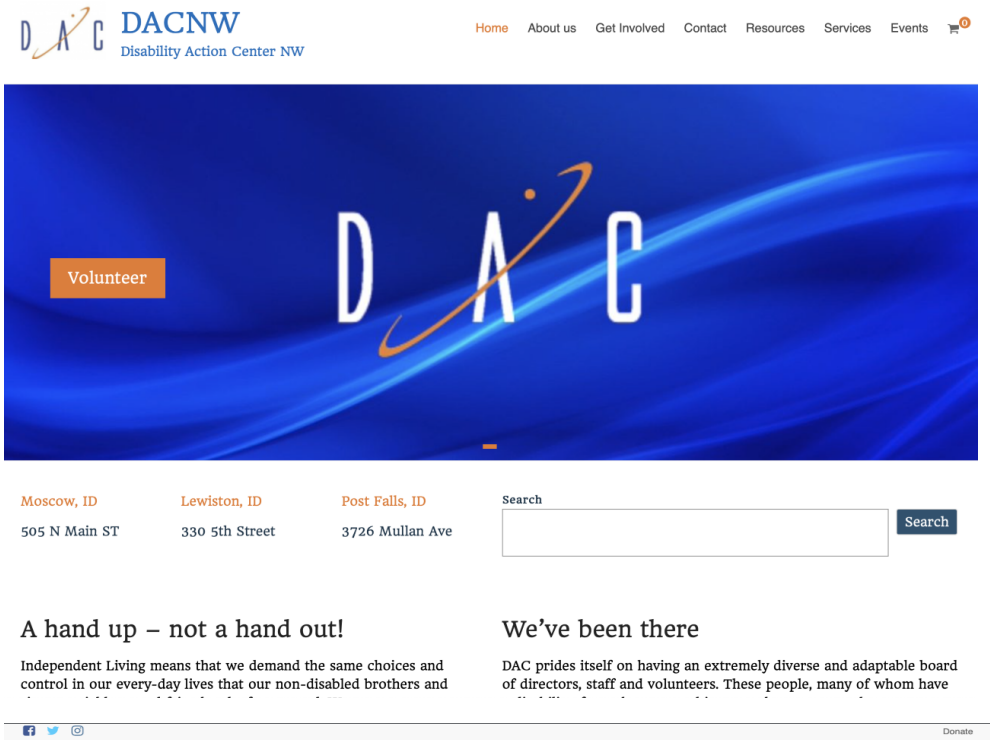
1. *Service name*: Backup
   Service provided from: UI Controller

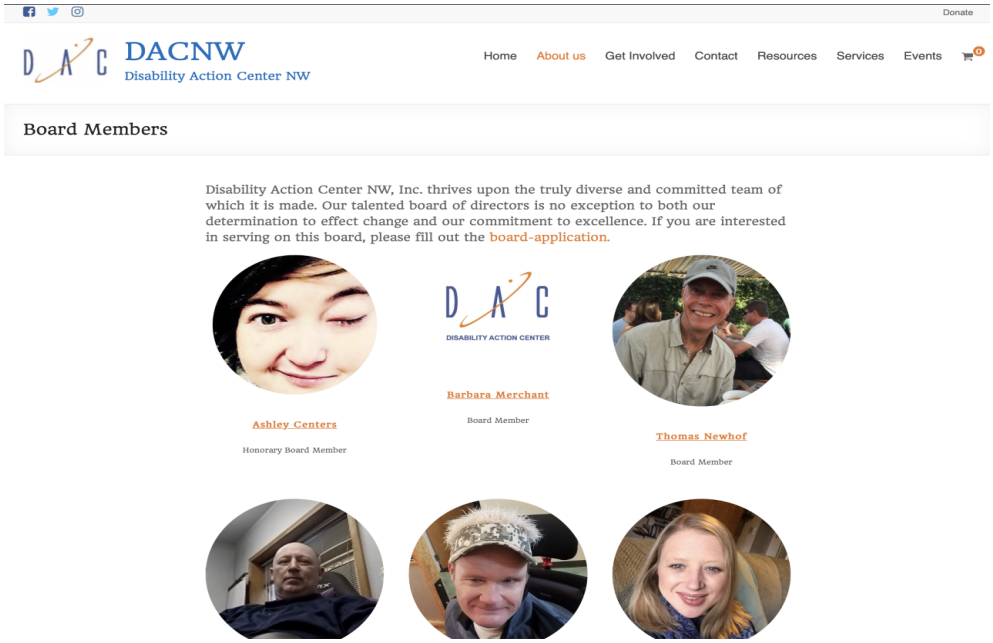
V.Data design

JSON datasets (events, news, meetings).The JSON format is not decided yet.

VI. 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):





The UI for Winform application is still in early stage and it doesn't have a complete UI yet. However, it should be simple with textboxes and a few buttons to modify the database.

VII. Testing and Acceptance Plans

VII.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..

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.

VII.2. Scope

This document discusses how we develop and test the INDEx project. The scope of this document covers our approach, required resources, the schedule of the testing activities (major work activities, products to be delivered, major milestones), and our plan and strategy for testing the product.

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.

2. Test plans

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.

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.

2.3. System Testing

2.3.1. Functional testing

This system testing plan will encompass the Index editing app pushing and retrieving data from the website. From the functional requirements section, this 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.

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.

2.3.3. User Acceptance Testing

The project will be tested several times, under 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.

VII.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.

VIII. Alpha Prototype Description

description

VIII.1. AWS API

VIII.1.1 Functions and Interfaces Implemented

get and put events and images

VIII.1.2 Preliminary Tests

basic functional system testing. (end to end)

VIII.2. Editor App Events

VIII.2.1 Functions and Interfaces Implemented

functions this that and the other

VIII.2.2 Preliminary Tests

tested this that and the other

VIII.3. [subsystem name]

VIII.3.1 Functions and Interfaces Implemented

implemented this that and the other

VIII.3.2 Preliminary Tests

tested this that and the other

IX. Alpha Prototype Demonstration

X. Future Work

asd

Glossary

INDEx: an organization which 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

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)