

Vizsite PRD

1. Goals and objectives
2. Target users and their needs
3. Main components and sitemap
4. Initial and future features
5. Non-functional but also import requirements
6. Wireframes
7. Potential risks
8. Analytics

Goals and objectives

Our goal is to make building and deploying customized static websites as simple as arranging the few blocks of lego. Users will drag and drop a few HTML elements and customize those elements using the CSS styles and deploy the website they have built with a single click. Vizsite will be a platform for small and medium businesses that want to build a website with few clicks.

Who's it for?

1. **Small and medium business** - Small and medium business owners who want to build their websites without needing a team of developers or monthly cloud costs.
2. **Beginner web developers** - Beginner web developers who are starting to learn the basic web technologies like HTML, CSS
3. **Web designers** - Designers who develop web applications
4. **Frontend developers**- Frontend developers who write frontend code using HTML or CSS

Why build it?

1. With the increase of internet adoption, every business now needs a website. but for building the website user needs to have an understanding of the web technologies which will take some time to learn. But using vizsite business owners are able to build their website without any prior knowledge. Which saves time and effort for them.
2. No code tools are on rising. No code tools like webtool, wix are being used by many people.

What is it?

User types

1. **Registered users** - people who have registered and can edit a website
2. **Non-registered users** - people who have don't have an account with us (Additional feature)

Website builder

A website builder is a place where the user will be building a site

a website builder will mainly contain 3 components

1. HTML Container
2. Editor
3. Style editor

HTML Container

1. HTML container will contain mainly two sections one is elements and the second one is components both elements and components are drag and droppable.

Elements are all basic HTML elements which are

1. <div>
2.
3. <p>
4. button,input,form,textarea

5. <h1>,<h2>,<h3>,<h4>,<h5>,<h6>
6. <table>,<tr>,<th>,<td>
7. ,,
8. <nav>,<main>,<article>

Components are repetitive website parts that can contain prebuilt HTML elements with CSS

1. Grid
2. Flex
3. Container
4. Form

Editor

1. visual editor

a **visual editor** is a place where the user will drag and drop the elements and components for building a website.

This is a droppable component.

Users should be able to select an item for applying the styles

Users should be able to remove an item

2. code preview

In **Code preview**, the user can visualize the code of the drag and drop items

Code preview should contain the copy button for copying the code

Code should be styled like in vs code

Code generation

We will generate the HTML and CSS code from the root object. We will loop through the root object and add elements into the dom.

3. website preview

In **Web preview**, the user can visualize the website and get an idea about how it will look like a website

4. Hierarchy preview

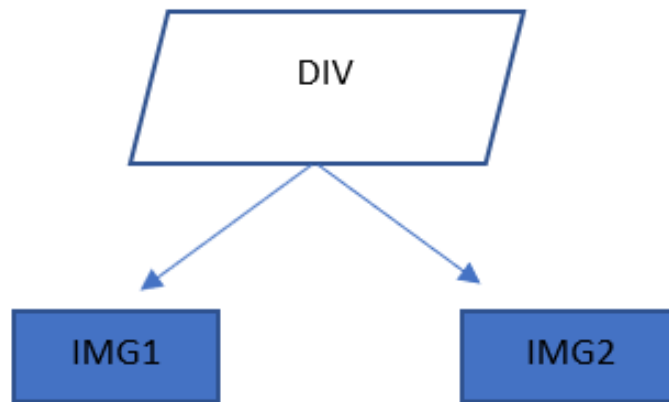
The hierarchy of elements like HTML tags will be predefined and will follow a tree-based structure where suppose if we add a tag and there will be the predefined rule where that tag can be moved within tags this confirms a strict hierarchy in code layout of each element implemented

- A Strict format of inter-user of elements is necessary to maintain the correctness of code.
- Integrity level of each element structure design
- An approach towards the more inclined UX experience
- Dynamic Nature of Elements of Website

Example-

```
<div style={styles.nav_buttons_container}className="nav_buttons_container">  
    
    
</div>
```

For Example for Below Html Code where the div area has two image holders, we will be having Dom Something similar to this -



Here, In Above Diagram we note points like -

- Every Node in this structure will be an HTML tag for the modal.
- Each Node will have a DOM property based on className.
- Lastly, Each node will have a CSS properties option for the same.

Style editor

Style editor contains all the CSS styles that can be applied to a selected item in the editor

Following are the CSS styles that we support

1. Color, Background
2. Margin, Padding
3. Border, Border Radius
4. Height, width, min-width,min-height, max-width, max-height
5. Box-shadow
6. Font-size, font-weight, text-transform
7. Position, display
8. Justify-content, align-items

Download code

Users can be able to download the code

Downloaded code would contain the single CSS and single HTML files

List of projects

list of all projects users has built or currently building.

Each project must contain

1. **Name** - Name of the project
2. **Started** - created date
3. **Last modified** - updated at the time
4. **Author** - Name of the person who created the project
5. **URL** - if the website is live URL of that website
6. Download

Create project

A new project user is going to create it will contain a form. the form contains the following things

1. Name required field

Competitors & Product inspiration

1. Webflow
2. Open chakra - No code builder which produces react and chakra UI code if we drag and drop the chakra UI elements.

Tech notes

Tech stack

React for Frontend,

firebase for authentication,

chakra UI for building the UI components,

react dnd for drag and drop functionality of the editor. Using this library we will the root object when using drag and drop and elements,

Node for backend

Why Firebase - Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users. Thus we chose this.

Firestore - firestore will be the db. why we are using firestore is we will be having all user data in

Netlify for frontend cloud service

Github for ci/cd

Element and component structure:

Every element or component would have the following structure

1. type of element
2. child items
3. styles

Type of element

Type of element is basic which type of element it is like div, span, or h1

Child items

list of objects which are direct child to this element

Styles

A style is an object that contains all the styles applied to this element.

Models

1. Projects
 - a. projectId
 - b. ownerId
 - c. projectName
 - d. url
 - e. ownerName
 - f. createdAt
 - g. updatedAt
 - h. siteId
2. Site
 - a. siteId
 - b. siteObj
3. User
 - a. id
 - b. name
 - c. email
 - d. mobile

Mobile responsive styles :

If user accesses the site from mobile, he will be able to view and download his projects, edit his basic info, but we will not provide access to editor as it needs a bigger space.

Future ideas

1. **Custom component create by user**
2. **Community component marketplace** - Users can build their own components and sell them

Deploy Site(additional feature)

Styles that are applied to a specific item should be saved and when we came back to the same elements applied styles should be shown in the style editor

Users can be able to deploy the site when he is done with the changes

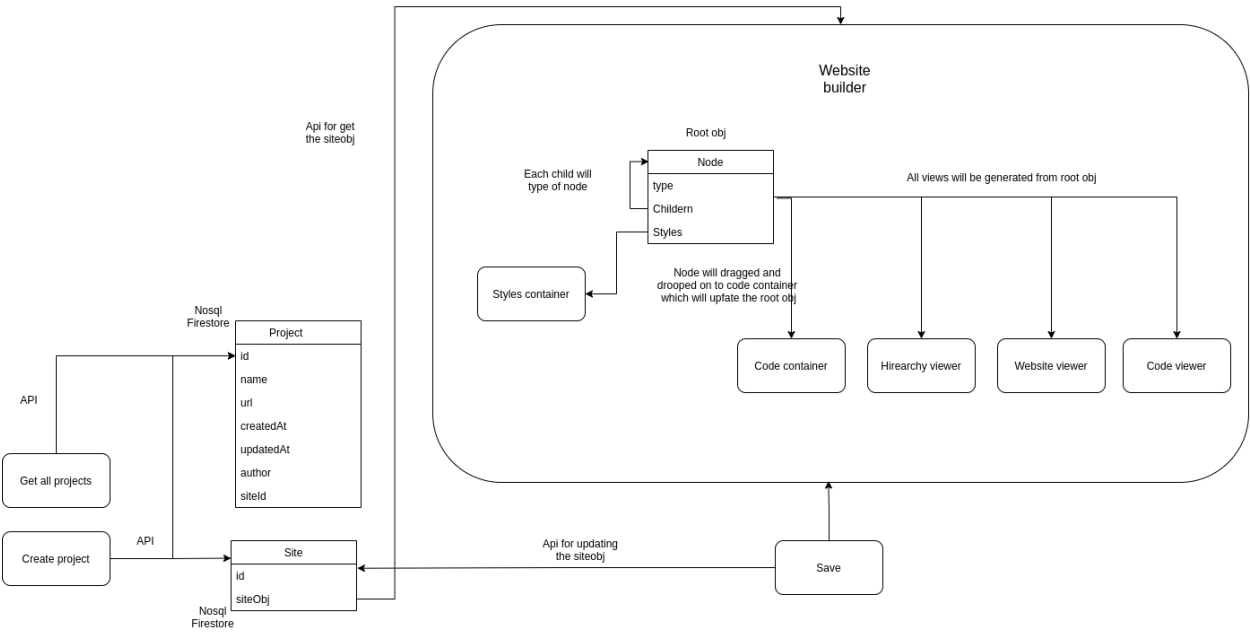
1. Users should give the unique domain name for which we will add our domain name and deploy it.

Glossary

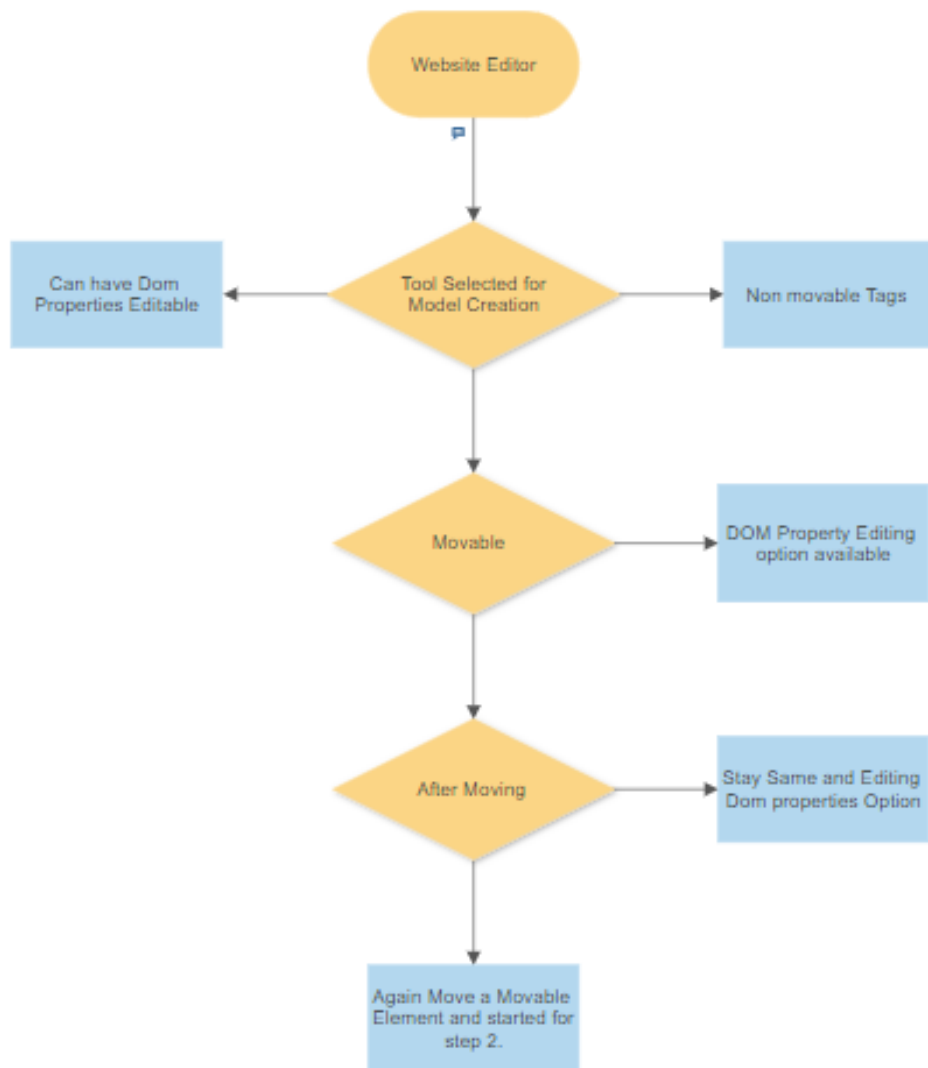
1. **Projects** - a single website
2. **Elements** - basic html elements like div,span,p
3. **Components** - repeated blocks of the website but not basic HTML elements like Grid, containers, Flex
4. **Styles** - Basic CSS styles like color, background, margin, padding,height
5. **Editor** - Where user will build the website
6. **Preview** - Can preview the website user is building
7. **Code** - Html and CSS code of the website user has built
8. **Subdomain** - A web address where users websites will be deployed
9. **deploy** - Making website user-created live on the internet
10. **Registration page** - input text fields for user to enter name(*), email(*), password(*), mobile number(optional)
11. **Login Page** - input text fields for user email, password, link to forgot password page. Also, we will have buttons for Gmail login, Facebook login, GitHub login
12. **Profile page** - user should be able to view and edit his/her profile information(name, mobile number, password)
13. **Projects page** - user should access his created projects and should be able to edit them
14. **Forgot password page** - input text field for the user to enter email id and button to submit. Once the user enters the email id and clicks on submit, we check whether the user has an existing account, if yes then we send an email to the registered email id to

reset the password. If the user does not have an existing account, we show an error saying "Provided email id is not registered"

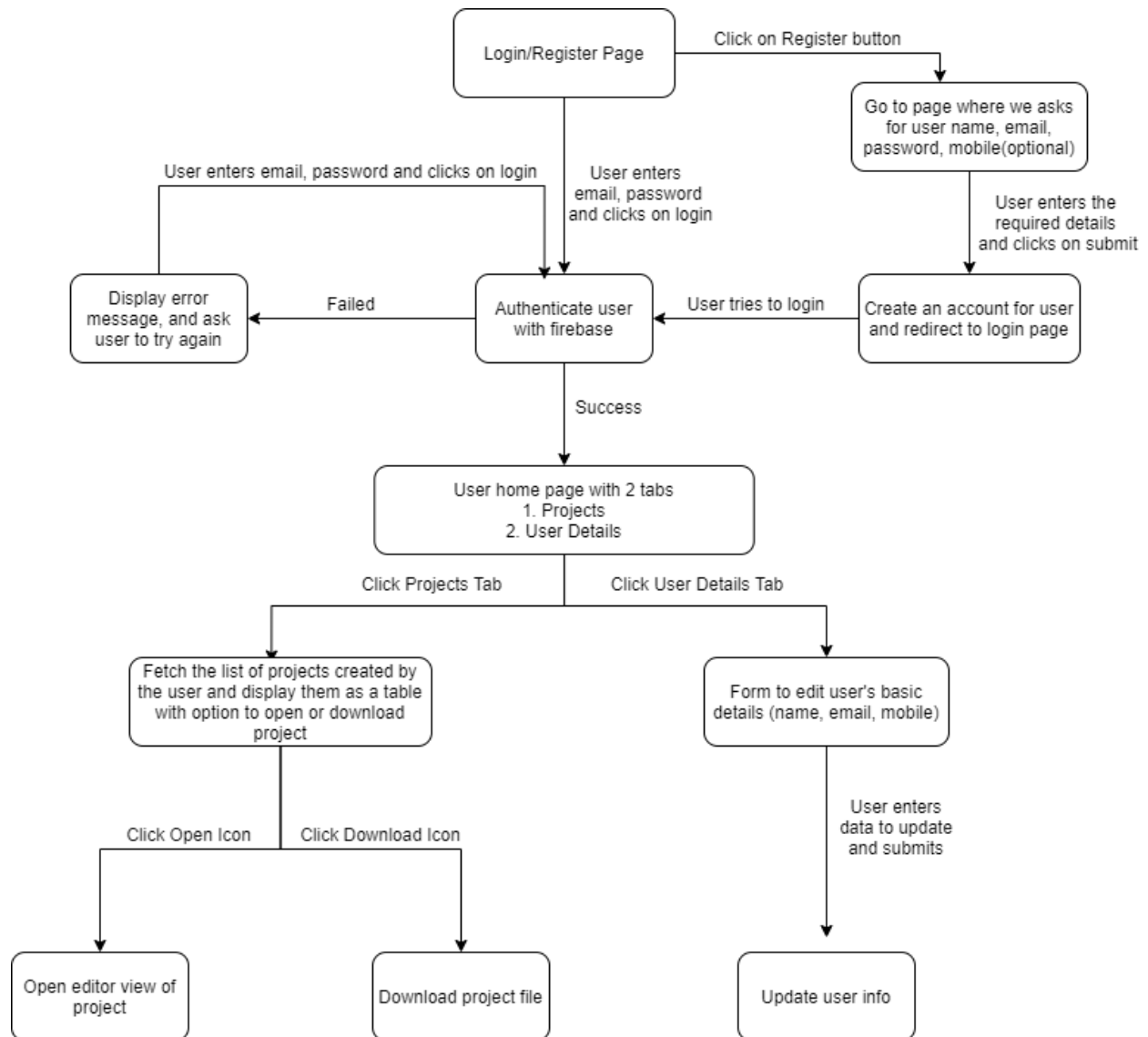
Editor flow chart

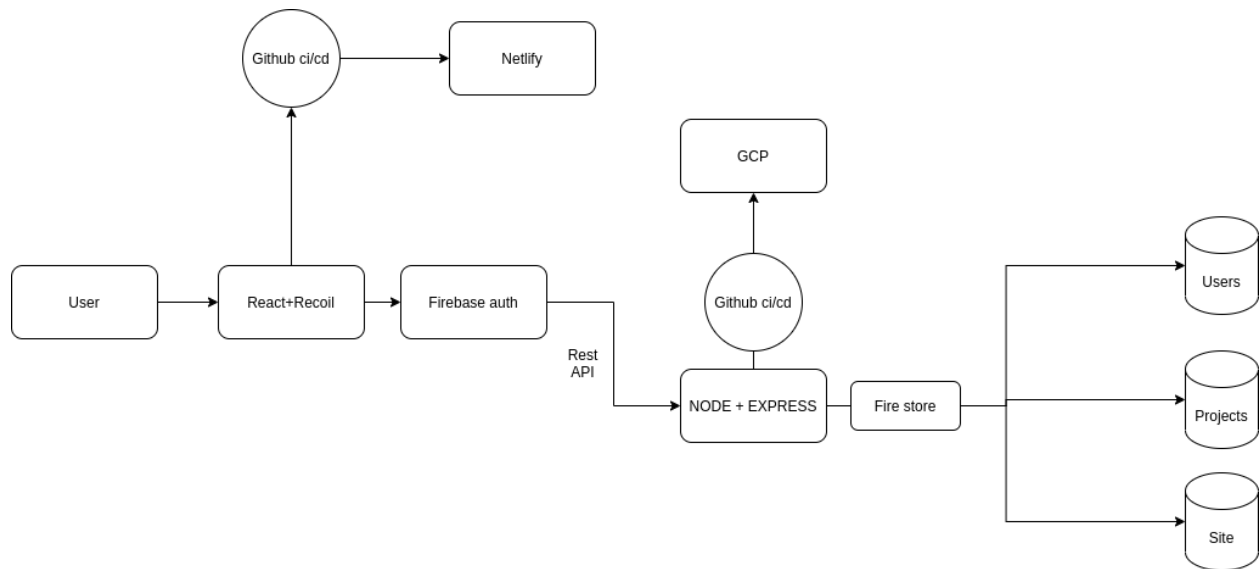


Hierarchy Design Flow:



Authentication flow:





Future ideas

1. Support mobile OTP login without password

We can register a user with a mobile number and OTP, once registered, the user can enter his/her details via the profile page. The user should be able to log in with a mobile number and OTP received at that time.


[Hierarchy of Elements Product requirements document.](#)

[Authentication And Authorization](#)

Timeline

Component timeline

Components	Time
<u>Authentication & Authorization</u>	3 days
<u>Dashboard</u>	1 day
<u>Create project</u>	2hr
<u>WebsiteBuilder - html container</u>	2day
<u>WebsiteBuilder - editor- container</u>	4 day
<u>Websitebuilder - editor - code - preview</u>	2 day

<u>Aa</u> Components	 Time
<u>Websitebuilder - editor - website - preview</u>	1 day
<u>Websitebuilder - editor - hierarchy - preview</u>	2 to 3 day
<u>WebsiteBuilder - styles container</u>	1 day
<u>Download</u>	