REPORT.md

Table of Contents

Analysis

Problem identification

With the internet constantly growing and more and more people relying on it, the demand for websites is continuously increasing. They can range in style from business portfolios and online stores to games. Regardless of who you are or what you do, a website is often expected of you, especially for businesses. However, the problem of creating a professional and functional website can be a daunting task for many individuals and small businesses, especially those who need more technical expertise or resources. Existing website builders may be challenging to navigate, require extensive coding knowledge, or lack the necessary design flexibility to create a unique and effective online presence. Furthermore, many website builders are geared toward specific industries or types of websites, making it difficult for users to find a platform that meets their specific needs. The task of manually programming it can seem very daunting.

The main aim of this project is to develop a website that allows clients to produce their website via a simple user interface, which alleviates the technical intricacies of HTML, CSS, and JavaScript and will enable them to focus on creating a website that reflects their brand and meets their business goals. Clients can select from various styles and themes (or create their own), upload media such as images and videos, and then interact with a drag-and-drop interface to organise a webpage. Each website they create would have a page dedicated to it, with options to customise the styles of the site, the ability to create, organise, and link together pages of the website, preview the website in a variety of display sizes, and add pre-made elements and edit the parameters of the elements in the site.

A drag-and-drop website builder that is easy to use is increasingly important in today's digital age, where having an online presence is crucial for businesses, organisations, and individuals. A user-friendly website builder that offers a wide range of customisable design options would greatly benefit users who may not have the technical know-how to create a website independently

Moreover, the rise of mobile devices and the increasing importance of responsive design have made it crucial for websites to be optimised for different screen sizes and devices. As such, this project would also be geared towards easily creating mobile-friendly websites to ensure optimal user experience.

The requirements for a client to be able to use it would also be low due to the entire application being contained within the website: this means that the client would only need a web browser and an internet connection. They do not need to install software onto their computer, nor do they need to worry about software updates.

In summary, the problem that this website builder aims to solve is to provide an easy-to-use, drag-and-drop website builder for individuals and small businesses, which can be easily customised to their specific needs and optimised for different devices without requiring technical expertise. This will greatly benefit users by allowing them to create a professional and functional website that meets their business goals and reflects their brand without needing a background in technology or coding.

Stakeholders

The clients for this solution could be a wide range of individuals or organisations looking to create a website without the need for technical competency or dedicating large amounts of time to it.

Some potential clients could include:

- Individuals: People who want to create a personal website, such as a blog, portfolio, or online resume.
- Small businesses: Owners or managers of small businesses who want to create a website for their business, such as an online store or service provider website.
- Entrepreneurs: Entrepreneurs who want to create a website for their startup or new business venture.
- Non-profit organisations: Non-profit organisations that want to create a website to promote their mission and raise awareness.
- Freelancers: Freelancers and independent contractors who want to create a website to showcase their work and promote their services.

The stakeholders are divided in two main groups: those who have some technical knowledge and can help with alpha-testing and feedback, and those who are not very technically literate, as that is the target audience of this solution. Having two groups will mean I get a variety of useful feedback when developing and beta-testing.

Stakeholder	Role	Technical Knowledge	Notes
Archie	Student	Medium	Will be able to help me with testing. Has some experience using website builders
Luke	Student	Medium	Will be able to help me with testing. Enjoys creating websites in his free time, well versed in usability and accessibility

---

Stakeholder	Role	Technical Knowledge	Notes
Sarah	Member of local Scout group	Little	Looking for a way to create a website without hiring a professional developer
Jake	Small business owner	Little	Interested in integration with social media
Kevin	Professional web hosting provider	High	Can provide insights into platform's compatibility with different hosting environments. Has insight into SEO
David	Barber shop owner and tech enthusiast	High	Excited to test features and provide feedback on capabilities and limitations

Why it is suited to a computational approach

This problem is suitable for a computational approach because it automates many technical and design tasks typically associated with creating a website. This is achieved by breaking down the website-building process into smaller, more manageable tasks that a computer program can easily handle.

For example, it can implement a drag-and-drop interface to allow users to easily add and arrange elements on a webpage, such as text, images, and videos, without coding. The website builder can also include a visual editor that allows users to easily customise the design and layout of their website, such as selecting from a variety of pre-designed templates or themes. The website builder can also apply pre-defined styles and formatting to the website and organise and link the pages together.

In addition, the solution can also use computational algorithms to optimise the website for different devices and screen sizes, ensuring that the website is responsive and easy to navigate on any device. This can be done by using CSS media queries, which can change the website's layout based on the screen's width, and JavaScript libraries that can detect the device and size of the screen and adjust the layout accordingly.

Using a computational approach can also ensure that the website is secure, fast, and reliable. It can use techniques like minifying, compression, and caching to make the website load faster and use authentication and authorisation mechanisms to ensure the website is secure.

The solution will be accessible through a server that hosts a website, which requires a computer to use. No alternative would not require a computer to be able to create a website, as at some point, the user will need to write and host the code that they have produced. This solution would act as a bridge between the client and the code, making it easy for users to create a website without needing to understand the technical details of coding.

Computational methods that the solution lends itself to:

Problem recognition

This solution is suitable for problem recognition because it addresses a specific problem that many individuals and small businesses face: the difficulty of creating a professional and functional website without technical expertise or resources. By providing a user-friendly, drag-and-drop interface, a website builder allows users to quickly create and edit their website without the need for coding or technical expertise. This addresses the problem of users not having the technical skills or resources to create a website themselves.

Furthermore, a website builder can also use computational algorithms to optimise the website for different devices and screen sizes, ensuring that the website is responsive and easy to navigate on any device. This addresses the problem of the rise of mobile devices and the increasing importance of responsive design in today's digital age.

Problem decomposition

The problem can be broken down into smaller tasks that must be programmed for the program to operate effectively.

- Creating an account system and a database to store the user's data.

> The account system would allow many users to access the server and edit their sites. The data would be stored in a server-side SQL database. The sites would be stored separately on the server.

- Creating a menu system for users to navigate to access different sites.
- Storing a list of template elements the user can preview and use in their sites.

> I need to decide how to store the template elements, display them to the user, and then implement them into a user's site while still allowing them to edit them.

- Creating a simple drag-and-drop interface that is easy to use and understand.

> This will probably be based on the grid-based positioning system that many existing website builders use or the constraint system that applications such as android studio use; however, the constraint system may not work with how HTML is created.

- Creating an effective way of storing the users' sites on the server.

> They would either be stored in HTM, CSS, and JS files, which would remove the need to convert them, or they could be stored all in XML files, which would make accessing the files easier: the program could convert the XML elements into HTML, CSS, and JS to allow it to be shown. This would allow for the storage of more information about the site and elements and would mean it could all be in one file, including all the separate pages.

- Converting the user's site into runnable HTML, CSS, and JavaScript so they can download and use it.

> There would need to be a way for JavaScript to do it so that the user can edit the site in the editor and a way for the server (written in Python) to do it as well

Divide and conquer

These smaller steps are all doable on their own, and combining them would make a divide-and-conquer approach. The advantage of it being coded in a modular way is that each part can be tested and built on its own without relying on other parts of the project.

Abstraction

The program uses abstraction as it removes the complex process of programming code by transferring it into a more straightforward, graphical interface - it provides a high-level interface for users to create a website while hiding the underlying implementation details. This removes the need for the client to have knowledge and experience in programming, opening the market to a much larger audience.

For example, the proposed drag-and-drop interface abstracts the implementation details of adding and arranging elements on a webpage using HTML and CSS code.

Initial talks with stakeholders

After meeting with each of the stakeholders and proposing the solution to them, I asked them each for feedback on what they would want the project to include, how they would want it to function, and, if they had technical knowledge, how I could implement the functionality. The questions I asked each person were:

1. "How much technical experience do you have?"
2. If they have technical experience: "Do you have any experience creating websites yourself, and if so how much?"
3. "Do you have any experience using existing website builders, and if so, what are your thoughts on them?"
4. "What are the ideal features that you would look for in a website builder?"
5. If they have a lot of technical knowledge: "What methods would you suggest when creating this project?"
6. "Do you have any other comments, ideas or requests about my proposed project?"

Archie, Student

1. Technical experience

> I study [A level] Computer Science and have used Python for a long time. The course teaches many various aspects of building applications and programming paradigms, which will be useful in the creation of a website builder.

2. Creating your own websites

> I haven't really used HTML, CSS, and JavaScript outside of small things here and there

3. Existing website builders

> A lot of website builders that I have seen before are very feature heavy and convoluted, which can often obscure the creation of a website. The paywall against many of their features often annoys me

4. Ideal features

> If I were looking for a website builder, I would want it to be easy to pick up how to use, and have it so that I could edit an existing site quickly - very user friendly. The ability to invite other people to help with the creation would be good, and the ability to import and export sites. I'd also want it to have a lot of customisability so that I can make the website look how I want.

6. Other comments

> Make sure to use VS code as its 100% better than everything else.

Luke, Student

1. Technical experience

> I mainly program in C (and sometimes Java), having learnt them while doing computer science. I've used web languages quite a lot.

2. Creating your own websites

> I find it fun to create websites with crazy premises in my free time. It helps me expand my knowledge and skills. Each one is better than the last. Recently I've been trying to implement more usability and accessibility requirements into the websites that I create, such as contrast ratios and ARIA tags.

3. Existing website builders

> I haven't really looked into website builders that much as I get more enjoyment from building them myself - its more of a hobby than a serious thing, and you have to pay for most builders. I have taken a look at bootstrap before, which is quite powerful.

4. Ideal features

> Well, you start with the drag-and-drop, and build from there. You want an easy way to position and reposition elements throughout the site - the ability to reorganise everything is really important. Styling customisability, so global styles and then the ability to change individual elements as well. Custom CSS and JS could also be a cool idea, but you'd need to look out for people putting in malicious code. Various templates that people can choose from is always a good idea, along with tags and filters for them. And accessibility: automatically assigning ARIA tags so that the user doesn't have to do it, checks to make sure the colors are high contrast enough, screen reader support. I'd be happy to help you with getting that stuff correct.

5. Programming methods

6. Other comments

> When I took a look at android studio, the way they do their positioning system is with constraints and XML formatted elements. That could be a cool way to do the positioning system, but it might also be a lot harder to program.

Sarah, Scout Leader

1. Technical experience

> No

3. Existing website builders

> The scout group is beginning to look into creating a website, so you've come along at the perfect time. In terms of what we've seen already, a couple of ideas have been put backwards and forward but we haven't come to a decision yet.

4. Ideal features

> I mean, as a scout group, we would want to be able to put up photos and videos of what we've been doing so that people are encouraged to join - we need to be able to change it quickly, and have multiple people be able to change it. We'd need to be able to change the color scheme so that it matches with the official scout branding. Sign up forms would be a feature we'd need so that people can sign up straight from the website. And the ability to add a privacy policy, so that it's compliant with our GDPR regulations. If I'm honest, a lot of us aren't very technically savvy, so it would need to be pretty basic.

Jake, Business Owner

1. Technical experience

> I did do some stuff at school, but it was only a little and quite a while ago

3. Existing website builders

> I've not really considered having a website for [the business], as it's quite small, and I've found that social media is good enough at the moment

4. Ideal features

> Well, the biggest thing that comes to mind is having my social media feeds on the front page so that people can easily get up-to-date information and see the latest posts quickly. I'd want a lot of styles to choose from so that I can make it look how I want it too - but not so many that I can't choose.

Kevin, Web Hosting Provider

1. Technical experience

> I work as a professional web hosting provider, making sure that clients have the best hosting plan based on requirements such as ease of use, security requirements, or expected lifetime

2. Creating your own websites

> My job doesn't really require me to, but I do often have to look through clients' code to ensure certain requirements are met, such as accessibility guidelines or creating suitable SEO information for the site. I do have knowledge of how the web languages work.

3. Existing website builders

> The company that I work for provide a website builder as part of one of the pricing plans, but I'm not involved in its programming. It functions fine, but there are a lot of things that I would change about it.

4. Ideal features

> The visual aspect of any program is key - if it doesn't look good, is too cluttered, or doesn't look easy to understand, it will put people off using it. Nailing the design is an important aspect of any project, and you should make sure to prioritise it. Another thing to consider is site speed - if it takes forever to update the website, or it lags when you move things around, it will make people not want to use it. Other than that, you've got the other customisation features, organising objects, styles, content management, SEO customisability, and so on.

5. Programming methods

> I'd suggest using Flask for this project, as you are already quite used to python, and it is easy to set up. The go to front-end languages would be HTML, CSS and JS. Testing isolated features throughout will make the development faster, and make it easier to link components together.

6. Other comments

> Make sure you talk to your other stakeholders throughout to ensure that the final project will be something they'll want to use - their input can be vital.

David, Barber Shop Owner, Tech Enthusiast

1. Technical experience

> I've done a lot in terms of technical stuff - I program a lot in my spare time to help with tasks, I have a pretty nice computer setup, a couple of raspberry pies setup around my house to run automated processes, and I'm big into collecting older tech.

2. Creating your own websites

> I've used HTML before. I wouldn't say I'm the best at it, but I know how it works. As for creating websites, I've not gone that far.

3. Existing website builders

> I tend to stay away from automated solutions to technical programming, especially online ones, where subscriptions and paywalls are a massive component of the industry. I'm beginning to look into creating a website for my shop, but I'm cautious about using website builders and may go down other avenues.

4. Ideal features

> The design is the most important bit for me - I should be able to create the entire site to look exactly how I want it, and then be able to release it to the public. Making sure that everything looks how it should, that precision is a big importance to me.

5. Programming methods

> You should definitely look at how other services have created their versions before, as it can give you inspiration and help you work out how you want to approach the construction of it. Make sure to be constantly testing and looking for feedback - I'd be happy to help with that.

Analysis from stakeholders

After gathering feedback from the stakeholders, it became clear that there are several key features that they wanted from a website builder:

- Easy to use and user-friendly interface
- Ability to edit an existing site quickly
- Ability to invite others to collaborate on the site
- Import and export sites
- Enough customizability to make the website look as desired
- Styling customizability with global and individual element options
- Custom CSS and JS options

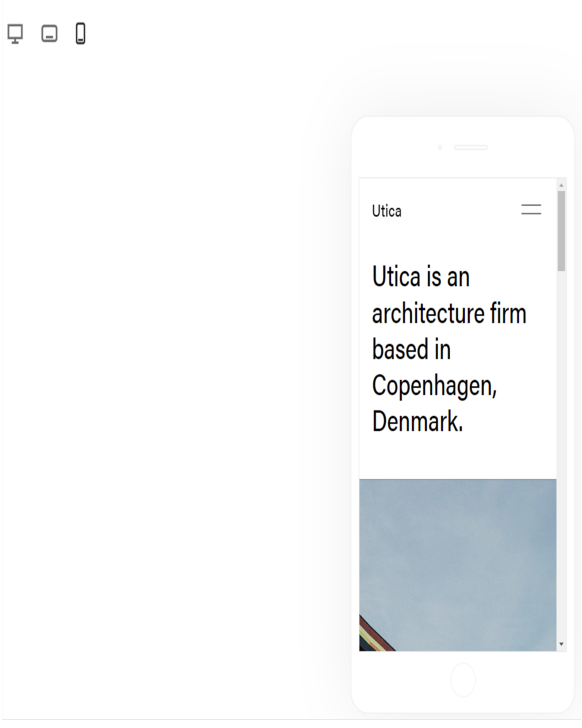
- Various templates with tags and filters
- Accessibility features, including automatic ARIA tags and contrast ratio checks for color
- Support for screen readers
- Ability to change color schemes to match brand identity
- Sign-up forms and privacy policy options
- Integration with social media feeds

It's also worth noting that some stakeholders expressed a willingness to help with the implementation of certain features, so it may be beneficial to involve them in the development process. Additionally, the website builder should aim to be basic enough for those with minimal technical experience to use, while also providing enough features and customizability for those with more technical experience to make use of.

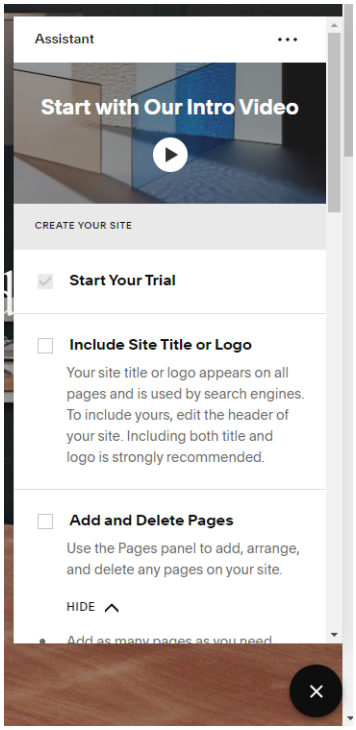
Research

Existing solution - Squarespace

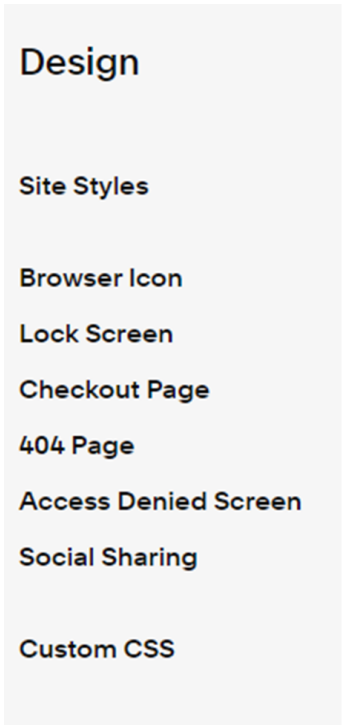
Their template list gives the option to preview the website, with all its functionality on a separate page. They allow their users to view it in different sizes as well.

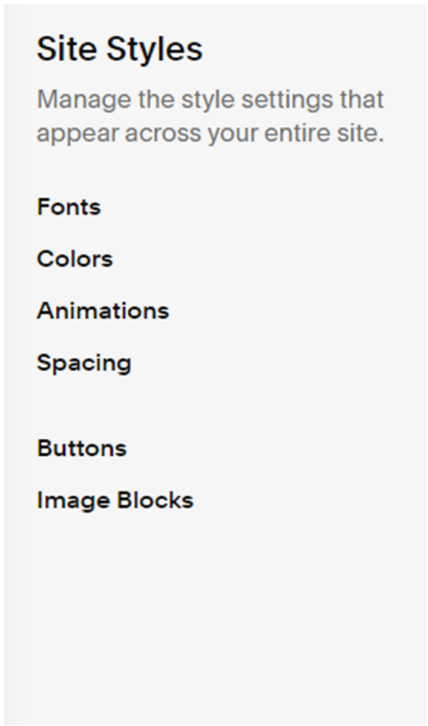


When first editing the site, Squarespace offers an assistant with some basic first steps to creating the website, making it easier for the client to understand how the editor works and how to use it effectively.



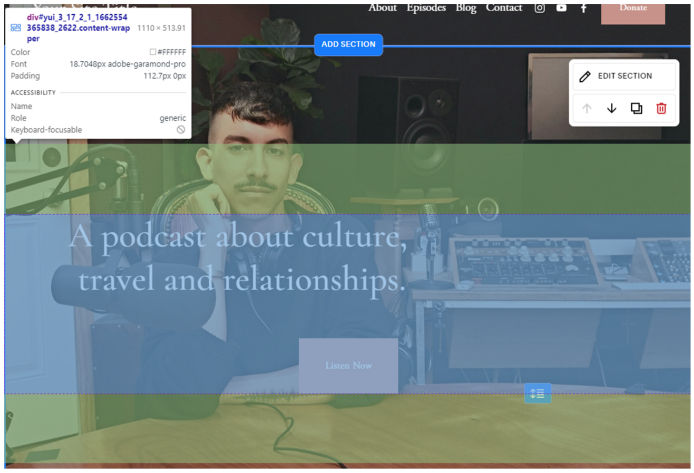
Their design options include styles, browser icons, 404 pages, and custom CSS. They can change the fonts, colour scheme, global animations, spacing, and default styles for certain widgets.

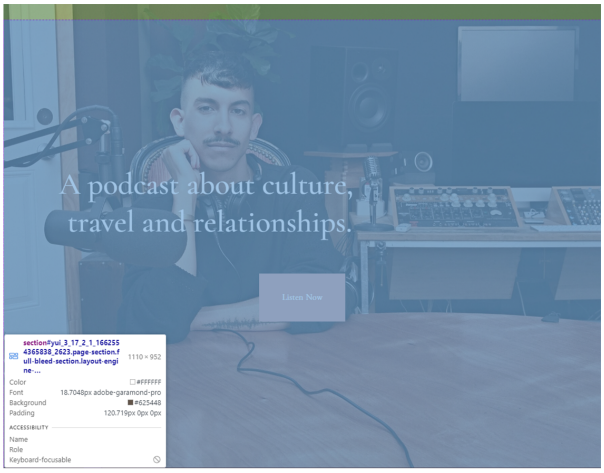




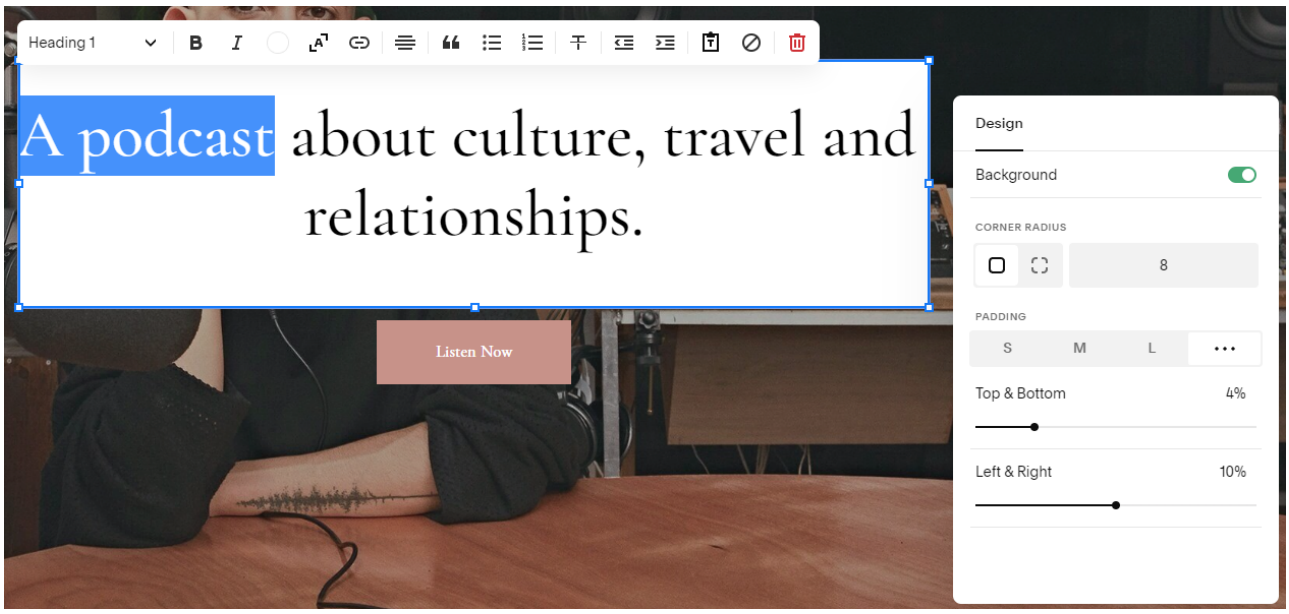
Their editor works in the conventional way of a grid-based system, where the user can place elements anywhere on the grid. It will then assign the item the style property `grid-area:row-start/col-start/row-end/col-end` or `grid-area:y/x/height/width` to define the position of the element. They have different attributes for different screen sizes, and the user can edit both styles separately by switching between laptop and phone modes.

The website is split into sections, where each section contains a content wrapper with the grid positioning system inside.





Selecting text gives the user a popup that displays the text formatting options. Whenever the user clicks on an element, they get a different popup that displays its design options.



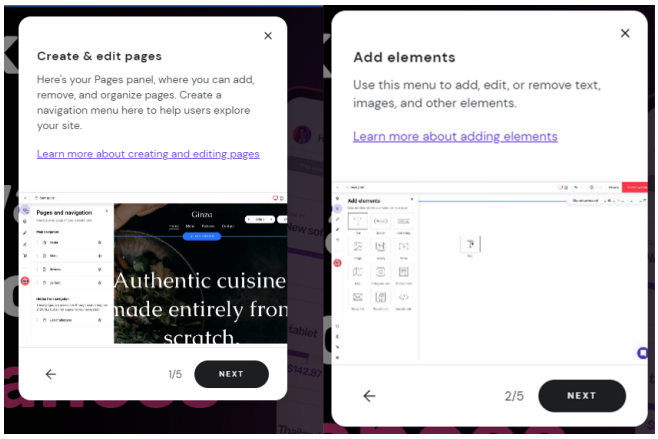
Parts I can apply to my project

This solution will have the option to preview and use templates similarly to how Squarespace does it, along with their grid positioning system, which is, for lack of a better phrase, an "industry standard." There will also be a similar formatting option setup, but it will be docked on the right-hand side with all the formatting settings in the same place.

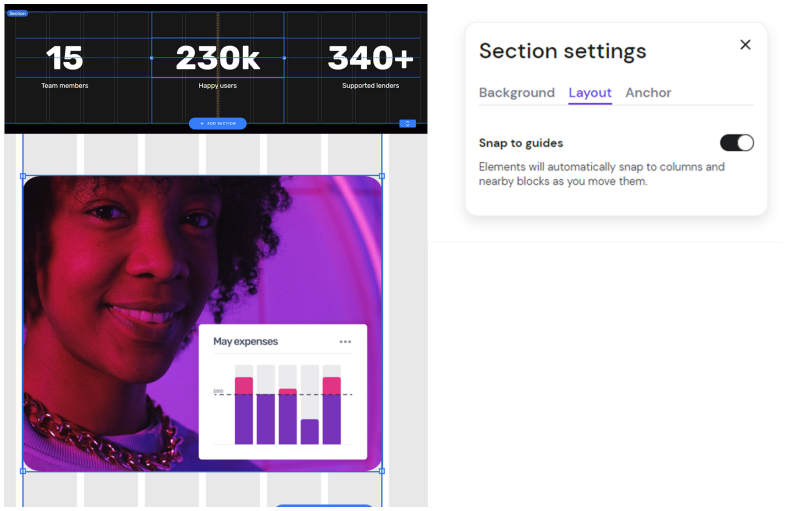
Squarespace also offers a variety of features such as e-commerce integration, SEO tools, analytics, 24/7 customer support, and a Content Management System (CMS) that allows users to update and manage their website's content easily. Although these features are helpful in a website builder, I intend leave them out of the solution's first release due to the timeframe for it and how long these features would take to make. This solution would have a CMS, but not at the scale or capability of Squarespace.

Existing solution - Zyro

Zyro also uses an assistant to help users understand how to use their editor.



Zyro has two ways of positioning objects; one is very similar to the way Squarespace does, with a grid positioning system, and the second is a smart layout. It instead uses only columns to position, and the elements can be moved up and down said columns freely and snap to other elements, like how many editors like photoshop might do. The user can toggle the snapping to other elements in section settings.



Something else Zyro does is have all of their style attributes defined in one class, which relies on variables such as `--grid-row`, `--m-grid-column`, and `--element-width` that are defined in `element.style` (the style attribute of the HTML object), which have been inserted with JavaScript.

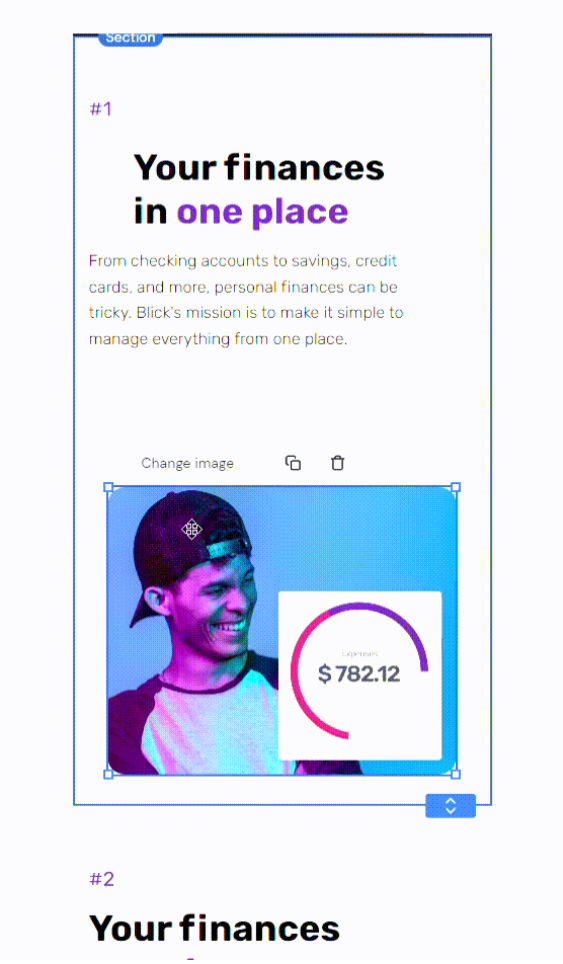
The CSS class with all of the variable references:

```
.layout-element {
  position: relative;
  left: var(--left);
  z-index: var(--z-index);
  display: grid;
  grid-row: var(--grid-row);
  grid-column: var(--grid-column);
  width: var(--element-width, 100%);
  height: var(--element-height, 100%);
  text-align: var(--text);
}
```

The HTML style attribute with all of the variable declarations in it:

```
element.style {
  --text: left;
  --align: flex-start;
  --justify: flex-start;
  --m-element-margin: 0;
  --z-index: 4;
  --grid-row: 3/4;
  --grid-column: 3/6;
  --m-grid-row: 2/3;
  --m-grid-column: 1/4;
}
```

Their image resizing system is good. It uses the `object-fit: cover` property in the style of the image and changes the width and height attributes when being dragged, as explained later.



When moving elements around the layout, Zyro adds four variables to the element, `top`, `left`, `width`, and `height`, which they use to render the element positioning while you are moving it. When the user releases the element, these values are removed. This positioning will be done in JavaScript by taking the position of the cursor when the user clicks on the element, getting the element's position when they click on it, and then offsetting the element's position by the amount they move the cursor. Then, when the user releases the cursor, it runs the code to calculate the new grid positioning of the element. They also have a max width for desktop mode, where the element cannot be moved further.

Parts I can apply to my project

Like Squarespace, Zyro uses a grid positioning system that will be implemented in this project. Their system for moving elements around with temporary `element.style` declarations is also a good way of implementing it that will be used as well.

Zyro also includes a built-in AI-based website creation tool called Zyro AI Writer, which can help users easily create website content. Similar to the Squarespace e-commerce system, this will not fit into the timeframe of the initial release of this solution, so it will not be implemented.

Key features of the solution

This solution will be a web-based, multi-user program where the user uses a grid-based, drag-and-drop system using pre-defined template elements that they can customise. There will be a tutorial for creating a site to help new users understand the system. The user will be able to customise styles for their site, organise pages, access a library of pre-defined templates for widgets such as text, buttons, or links, and control the styling of each element in their pages.

The aim is to have an easy learning curve and a low entry bar for understanding so that anybody can use it. Ease of use is also crucial - the cognitive load on the user should be low enough that it allows them to focus on designing the website due to

Limitations of the solution

The main limitation is that, as a server-side application, the user will always need an active internet connection to access it, and if the server goes down, there will be no way of using the program.

Hardware and Software Requirements

Hardware Requirements

The client will need a computer capable of accessing the internet.

In terms of the server side, hardware requirements may include the following:

- A server to host the website builder application and handle user requests. This server could be physical or virtual and run on any operating system that can also run Python (as it is the language I will be using to program the backend).
- The server will require a CPU with enough processing power: it may need a high number of cores and a fast clock speed to handle high traffic and many concurrent users.
- The server will need enough memory (RAM) to handle the requests and processes of the application.
- The server will require sufficient storage to store the website builder application and the images and files uploaded by the users to the CMS.
- The server will need a high-speed network connection to handle incoming and outgoing traffic between itself and the users. The faster the connection, the better the user's experience will be.

In addition to the above hardware requirements, it should be designed with scalability in mind so it can be adjusted if the application size or the number of users increases.

Software Requirements

The client will need a JavaScript-compatible web browser and an active internet connection.

For the server, software requirements may include the following:

- The operating system used on the server should be able to run Python alongside other software listed below.
- Python - it will need to be able to run Python in order to host the website builder.
- A file structure system to store the code and assets for the website builder.
- A database or file structure system to manage and store the data and media uploaded by the users to the CMS.
- Firewall software to monitor incoming and outgoing traffic and stop potential threats from corrupting the system.

- Load balancer software may be required when the user base increases to manage server resources and route traffic to available servers.

Success Criteria

Essential Features

- Login system
- the ability to view the password with the all-seeing eye
- Signup fields to be name, email, username, and two passwords to make sure they get it correct
- SQL database that stores user and site information
- fully functional error checking on all fields as follows

> - All fields must not be empty
> - Name can have spaces and non-alphanumeric characters and must be longer than 2.
> - Email must be in an email format.
> - Username cannot have non-alphanumeric characters and must be longer than 2.
> - Password must be longer than 8.
> - The repeated password must be identical to password.
> - Email cannot already be in the database.
> - Username cannot already be in the database.

- The homepage, when there are no sites, displays a prompt to create a new site
- The homepage, when the user has created sites, lists all of them along with a "create new site" button
- When creating a site, the user will get the following options

> - Website Name: at least four characters, and any illegal characters are converted into dashes. The user is given a preview of what their site name will look like when it does not match the criteria.
> - Description: optional
> - Whether the site is public or private: determines who has access to the site URLs

- Sites can be accessed with the URL: `/<username>/<sitename>`, and, if public, can be viewed (but not edited) by anyone from this URL. If private, other users will be told this and redirected home.
- The site will have a config file, where it stores all of its global variables - mostly style choices - which have been selected when creating the site. These can also be edited at any time on the site's homepage.

> These variables include primary, secondary, accent, and grey colours, primary and secondary fonts, and animation types.

- File storage system to store user site files
- CMS system for users to be able to upload custom content

- The site page (`/<username>/<sitename>`) can be programmatically assigned due to the Python backend: it can take both parameters, search for them in the database, make sure that the current user has permissions, and display the appropriate site.
- On the site page, the user will get a preview of the website, along with customisability options for the website: the ability to edit the site, reorganise the site structure (which pages go where), edit site settings (such as default colours), and export the site.
- When editing the site, the organisation will look like this

> - A navigation bar on the left that contains the options: "website pages", where you can navigate to a different page, "add section", where you can add another template section to the current page, "website styles", where you can change global settings such as fonts and colours, and "add element", where you can drag and drop individual elements into the canvas to edit.
> - A central canvas where the actual web page can be previewed
> - A popup modal that appears when the user needs to add an element or section
> - A styling section on the right-hand side where the user can edit all of the styling properties for a selected element

- The central canvas will import the raw HTML and CSS files from the server and rely on data tags in the HTML element to understand what does what and how to edit it.
- Whenever a widget is selected, a box will be drawn around it, with the ability to resize it. The style menu on the right will also populate with style options for the selected element that can be changed in real-time and previewed when hovered over so that the user can easily understand what certain buttons will do.
- Whenever a widget is selected and held, an outline of the parent section's grid system is previewed, and the element can be moved around. It does this by tracking the cursor's position and relating that to the start position of the cursor on the widget (the anchor point) to render it in the correct place using left, right, top, and bottom CSS tags. When released, the widget will snap into the nearest grid space to where it was released. A similar thing happens when the user selects and holds one of the resize elements on the outline, where it tracks the cursor and then snaps into the closest grid space to resize it.
- The position parameters that are changed, as described above, are separate for the desktop and mobile views of the web page. Changing the position when the page is in desktop mode will not affect the position in mobile mode and vice versa.
- When a widget is right-clicked, it will show useful commands such as copy, paste, delete and duplicate.

Desirable Features

- Ability to (export and) import sites in a zip file so you can transfer them between sites, which is different to downloading a usable copy of the website. An export function may not be necessary as it is given in the site settings.
- A list of predefined templates for sites when creating a site, that can be organised and filtered via relevant tags.
- When creating the site, the user can select options that allow them to change the default styling properties of the site.

> These will be the options for color palettes and font families.

- The site owner can assign other users the ability to edit public or private sites, but there cannot be two people editing simultaneously.

> This would be achieved by modifying the database structure to have a linking table between the User and Site tables. This is explained later in the design section.

- Accessibility and support: Automatic ARIA tag assignment inside website builder, accessibility analysis tools to ensure that the current color palette is acceptable, and adequate support for screen readers in created websites.
- Audience interaction features such as forms, social media feeds, and article posts.
- To export the site, the user will have two options that will be clearly defined in the UI
- They can download the site, which will download a zip file containing all the required HTML, CSS, and JavaScript code, so that they can unpack the archive and run the webpage by simply opening the HTML file.
- They can export the site, which will download a different zip file that contains all of the internal files that Kraken uses to run the editor for the page. This means the user can download backups and send their websites to others.