General Documentation

Table of Contents

[**Introduction** 4](#_Toc68443209)

[**Updating the Documentation** 4](#_Toc68443210)

[**1. Major Releases** 4](#_Toc68443211)

[**2. Change in the Design and Development Team** 4](#_Toc68443212)

[**Using Documentation** 4](#_Toc68443213)

[**App Flow** 4](#_Toc68443214)

[Index.js File 4](#_Toc68443215)

[App.js File 5](#_Toc68443216)

[Container Files 5](#_Toc68443217)

[Component Files 5](#_Toc68443218)

[Component Files Technical Docs 6](#_Toc68443219)

[Home Component 6](#_Toc68443220)

[Overview 6](#_Toc68443221)

[**Variables** 7](#_Toc68443222)

[Slider 7](#_Toc68443223)

[UseHistory 7](#_Toc68443224)

[**Home Component File** 7](#_Toc68443225)

[incubatorClicker 7](#_Toc68443226)

[eventClicker 7](#_Toc68443227)

[Settings 7](#_Toc68443228)

[Swayam E-Cell Incubator 8](#_Toc68443229)

[Swayam E-Cell Events 8](#_Toc68443230)

[Swayam at a Glance (Photo-Gallery) 8](#_Toc68443231)

[Swayam in Numbers 8](#_Toc68443232)

[Testimonials 8](#_Toc68443233)

[Footer Component 9](#_Toc68443234)

[Overview 9](#_Toc68443235)

[Variables 9](#_Toc68443236)

[nameRegularExpression 9](#_Toc68443237)

[emailRegularExpression 10](#_Toc68443238)

[numberRegularExpression 10](#_Toc68443239)

[subjectRegularExpression 10](#_Toc68443240)

[messageRegularExpression 10](#_Toc68443241)

[The footer component file 10](#_Toc68443242)

[Address Part 10](#_Toc68443243)

[Form Component 11](#_Toc68443244)

[Footer Nav-Bar 11](#_Toc68443245)

**Introduction**

This documentation is prepared by the Swayam Web Design and Development team for future reference on the work performed in Q2 and Q3, 2021. The documentation clearly states how a developer or a designer can change the design, update the development process, and change the documentation to reflect the same.

Developers must follow the rules mentioned in this documentation. The rules may be changed or updated to suit the industry developments and the requirements. All the changes must be published in the documentation to act as a reference for future teams.

**Updating the Documentation**

**1. Major Releases**

The documentation must be updated after every major release. Moreover, at least two copies of documentation (both the old and the updated versions) must be maintained to prevent any mishaps.

1. What constitutes a major release?
2. Addition of breaking features.
3. Breaking changes in security and usage policies.
4. Major changes in the design (more than 50%).
5. Addition of more than 10 third-party libraries or modules.

**2. Change in the Design and Development Team**

If any developer that is assigned a specific task resigns before completing that task, they must enter the details on how to change the code and the work progress in the “pending” section.

**Using Documentation**

The documentation will have multiple sections. Each section refers to a specific feature or an event. At the moment, there are two sections in this documentation: General and technical documentation.

**App Flow**

The application contains only one HTML file. Using Single Page Web Applications, we can drastically improve the user experience. The HTML file is then modified using JavaScript. This is the essence of an SPWA.

### Index.js File

When a user opens the application, the ‘index.js’ file will be the first one to load. The file directly changes everything in the body to what was mentioned in the JavaScript file.

1. ReactDOM.render(
2. <React.StrictMode>
3. <App />
4. </React.StrictMode>,
5. document.getElementById('root')
6. )

The ‘App’ HTML tag in the above code refers to the App.js file. The App.js file is present in the ‘src’ directory. You can locate the App.js file in the ‘src’ directory. To use a container as an HTML tag, it must return JSX.

### App.js File

The App.js file should be used as a default routing file. All the routes must be mentioned in this file. Any route that is not mentioned in this file will not be rendered in the application. The application is currently taking advantage of ‘[react-router-dom](https://reactrouter.com/web/guides/quick-start). This module contains the It contains the DOM bindings for React Router.

Example usage:

1. <BrowserRouter>
2. <div **className** = 'App-Body-Div'>
3. <Switch>
4. <Route **path**='/' **component** = {HomeContainer} **exact** />
5. <Route **path**='/incubator' **component** = {Incubator} **exact** />
6. </Switch>
7. </div>
8. </BrowserRouter>

React router provides multiple methods such as ‘Route’ and ‘Switch’ to declare our routes. The <Switch> tag works just like the switch statement in JS. The Route tag takes multiple arguments: the path argument is the URL path; the component attribute contains the container that the app must render on that page. Finally, the ‘exact’ attribute specifies that the app must render the mentioned component only when the path matches completely and not just partially. (Remove the exact statement for the ‘/’ path to see the importance of this attribute)

### Container Files

They container a reference to the component files. Moreover, both the Navbar and the Footer are added to this file. Container files are mostly declared as classes. They can handle the state without using hooks.

### Component Files

Component files contain the main JSX that is rendered on the website. Changing JSX in this file will change the website.

# Component Files Technical Docs

## Home Component

### Overview

The Home Component is the main page of the website. It consists of information regarding Swayam and its activities.

**NAVBAR COMPONENT**

**Swayam**

The ***Entrepreneurship*** cell of Vasavi College of Engineering

**Incubator Container**

**Events Container**

[**PHOTO GALLERY**](#_Swayam_at_a)

[**Swayam data**](#_Swayam_in_Numbers)

[**Testimonials Slide**](#_Testimonials)

[**FOOTER COMPONENT**](https://d.docs.live.net/2a443dc89a828fe3/Documents/Footer%20Component.docx)

# **Variables**

### Slider

The Slider is imported from ‘react-slick. This tag allows us to display carousels on our website. We can do it using vanilla JS, but manual coding of such design elements is considered inefficient. Further information on what Slider is and what parameters it takes can be found in the [official react-slick docs](https://react-slick.neostack.com/).

Note: This is a lazy import. [Lazy loading](https://developer.mozilla.org/en-US/docs/Web/Performance/Lazy_loading) improves the performance of an application drastically.

### UseHistory

[UseHistory](https://reactrouter.com/web/api/Hooks/usehistory) is a React Hook that gives us access to the history instance. We can use this history instance to go to a new page or the old page.

Usage example:

1. var history = useHistory()
2. history.push("/incubator")

# **Home Component File**

It is a functional component that takes one argument ‘**props’**. The name of the argument can be changed, but it a good practice to name it as “props” because It will be easier for newbies of the team to understand the code.

JavaScript allows us to write **Nested-Function** (function inside a function). We implement the same in the functional Component to handle change on the webpage.

### incubatorClicker

This is used to handle click events, when the user clicks the incubator container on the home page, they are redirected to the incubator page. This is an **onClick handler**.

### eventClicker

This is another **onClick handler** used to handle click events on the events container on the home page. The users are redirected to the events page when clicked on the box.

### Settings

This variable contains all the settings to configure the testimonials carousel to run.

### Swayam E-Cell Incubator

It is the link to the incubator page. Users are redirected to the incubator page when clicked on the container.

### Swayam E-Cell Events

It is the link to the events page. Users are redirected to the events page when clicked on the container.

### Swayam at a Glance (Photo-Gallery)

This container contains the images from Swayam events. All the images are set as background images in the homeComponent.css file.

### Swayam in Numbers

This container contains data for the total number of events hosted, the number of startups incubated, the number of students impacted, the total fundraised by the board members. All the data is hardcoded in the “homeComponent.js” file, and **not** rendered dynamically.

### Testimonials

This container consists of the information on various start-ups incubated by Swayam. The slider consists of two testimonials in a single slide. If the testimonials are less than 2, then it simply acts as a container.

‘react-slick’ is used to make the slider. The ‘[**settings’**](#_Settings) variable configures the slider.

## Footer Component

### Overview

The footer components contain a contact form, the location of the college, and a navbar.

Graphical user interface

Description automatically generated

## Variables

All the variables are declared as ‘constant’ i.e., cannot be changed at runtime.

### nameRegularExpression

This variable is a regular expression for the user handle. We use it to check the format of the name on the contact form. The user handle can contain any number of **characters** between 3 and 15. It cannot start with a blank space, though the user can add space anywhere else in their name.

### emailRegularExpression

This variable is a regular expression for the user’s email. We use it to check the format of the name on the contact form. The user’s email id must start with an alphanumeric character (local part of the address), followed by the address sign and the domain name. The local part of the address may contain periods(.), underscore (\_), or hyphens (-). The domain name should be at least two characters long and can span to any number of letters.

### numberRegularExpression

This variable is a regular expression for the user’s phone number. We use it to make sure the user enters only numbers in the given field. The number’s length can be ten or eleven (if the user enters the country code).

### subjectRegularExpression

This variable is a regular expression for the subject field on the contact form. We use it to make sure that the user does not leave it blank. The user is allowed to enter any alphanumeric character, period (.), question mark (?), underscore (\_), hyphen (-), exclamation mark (!), comma (,), forward-slash (/), backward-slash (\), angle brackets (<>), colon (:), semi-colon (;), curly brackets ({}), round brackets (()), square brackets ([]), asterisk (\*), ampersand (&), hash (#) and spaces. The subject can’t be blank and should contain at least three characters and at most fifteen characters.

### messageRegularExpression

This variable is a regular expression to validate the message field on the contact form. We use it to make sure that the user enters a valid message in the field. The user can enter any alphanumeric character, period (.), question mark (?), underscore (\_), hyphen (-), exclamation mark (!), comma (,), forward-slash (/), backward-slash (\), angle brackets (<>), colon (:), semi-colon (;), curly brackets ({}), round brackets (()), square brackets ([]), asterisk (\*), ampersand (&), hash (#) and spaces. The subject can’t be blank and should contain at least three characters and a maximum of hundred characters.

### The footer component file

The footer component is a functional component, which returns the JSX for the footer. It is further divided into the address component, the form component, and the footer navbar component.

The function ‘footerComponent’ takes one parameter ‘props’ as an argument. It is used to refer to the JSX attributes passed from the class Components.

For instance: In the Footer.js file, the form handler is passed as JSX attribute and we use it in the footerComponent.js file using props.formhandler

Now, let us look at the other parts of this component.

### Address Part

The address part of the component contains a map, address, and social media links. The map is embedded using the iframe attribute, with the source as the college's address.

### Form Component

The form part of the component contains a contact form with name, email, phone number, subject, and a message as its field. Each field has a change handler that handles the change on the input text field. The error and helper text attributes in each are set based on the regular expressions. If the entered details do not match the format specified in the regular expression, the error field becomes true, and the helper text changes to the error message.

### Footer Nav-Bar

The footer navbar part of the component has some links to change pages on the website. There are six links for six pages.

Each page has a description in a separate document.