



# Edupreneurs - A Digital Platform

Developing a digital business ecosystem in the Southern African province to leverage the EdTech infrastructure and Education software

Jannaten Nayem

BACHELOR'S THESIS  
December 2021

Degree Programme  
Software Engineering

## **ABSTRACT**

Tampereen ammattikorkeakoulu  
Tampere University of Applied Sciences  
Software Engineering

**AUTHOR:** Jannaten Nayem

Edupreneurs – A Digital Platform, developing a digital business ecosystem in the Southern African province to leverage the EdTech infrastructure and Education software

Bachelor's thesis 44 pages  
December 2021

---

This thesis presents aspects of the Edupreneurs platform, which builds a business environment for education in the Southern African regions. Covid-19 has revealed flaws in global innovation and entrepreneurial innovation. In the first section, the objective is primarily to learn more about the platform and, as a result, to determine what problem the platform is trying to solve. This study also explains more about React technology and how it can quickly fix it. At the end of the study, there will be a discussion about improving the platform to maintain it in the long run.

---

Key words: edupreneurs, react, bootstrap, api, layout, business, family, partnership, southern african, glowdom, schools, organizations, companies

## CONTENTS

1	INTRODUCTION .....	5
2	ABOUT EDUPRENEURS PLATFORM .....	6
2.1	Building blocks of Edupreneurs .....	6
2.1.1	For education partners .....	6
2.1.2	For companies and organizations partners .....	6
2.1.3	For families and stakeholders .....	7
2.2	Brainstorming of Edupreneurs .....	7
2.3	Strategy of Edupreneurs .....	8
2.3.1	How Edupreneurs collaborate .....	8
2.3.2	Business training of Edupreneurs .....	8
2.3.3	B2B vs B2C.....	9
3	REACT TECHNOLOGY .....	10
3.1	React components.....	10
3.2	React DOM and React DOM server .....	10
3.3	State vs Props .....	11
3.4	React lifecycle hooks.....	11
3.5	Event handling .....	12
3.6	Logical rendering .....	12
3.7	Stateful vs Stateless component.....	13
3.8	React hook and its uses .....	14
3.9	Reusable components.....	15
4	UP AND RUNNING WITH EDUPRENEURS .....	16
4.1	Installation of React app.....	16
4.2	Choosing third-party libraries .....	16
4.2.1	Axios (v0.21.1).....	16
4.2.2	Http-Proxy-Middleware (v2.0.0) .....	17
4.2.3	SWR (v0.5.6) .....	17
4.2.4	React-Router-DOM (v5.2.0).....	18
4.2.5	Bootstrap (v4.6.0).....	19
4.2.6	React Bootstrap (v1.5.2) .....	19
4.2.7	React Bootstrap icons (v1.4.0).....	20
4.3	Creating file and folder structure .....	20
4.4	Fetching REST API via SWR .....	21
4.5	Typography and Colours choice .....	22
4.6	Iconography of Edupreneurs.....	22
4.7	Wireframing of Landing Page.....	23

4.8 Desktop vs Mobile View in action .....	24
4.9 Responsive App by using Bootstrap and React Bootstrap .....	27
5 FUTURE IMPROVEMENTS .....	34
5.1 Solving the State Management issue .....	34
5.2 Implementation of Context API .....	35
5.3 Testing performance by using Lighthouse.....	36
5.4 Improving application performance.....	38
6 DISCUSSION .....	41
REFERENCES .....	42

## **1 INTRODUCTION**

Sharing information, building capacity, collaborating, providing business training, and educating company and organization partners is not easy in the Southern African region. (Sebulon. n.d.) Edupreneurs solve these issues. It gives the user opportunities for learning, doing business, and research. This platform was created with the popular library React, released by Facebook in 2011.

## **2 ABOUT EDUPRENEURS PLATFORM**

Edupreneurs, a digital business ecosystem, drive Southern Africa's EdTech infrastructure and educational software development. It attempts to deal with issues resulting from the lack of digital infrastructure and staff with digital skills. In particular, when the COVID-19 outbreak occurs in a pandemic and is severely affecting daily activities, the value of EdTech infrastructure cannot be underestimated. There are a few practical EdTech solutions in Southern Africa. However, such efforts are still isolated, limiting the region's success potential. This section briefly demonstrates the overview of the application.

### **2.1 Building blocks of Edupreneurs**

Education, business, organizational partners, and the end-user who views all the information are the building blocks for Edupreneurs.

#### **2.1.1 For education partners**

It is crucial to have the right platform for families to find schools in the Southern African region. The Edupreneurs platform helps the educational partners provide the schools' names, places, academic fees, academic contact details, many images, and a detailed explanation. Thus, families can determine what the school offers and what they can access to help them find their desired school (Leite, L. 2021a).

#### **2.1.2 For companies and organizations partners**

Companies and organizations in the Southern African region are rapidly expanding. Many corporations spend a large amount of money on organizational construction. That is why the Edupreneurs platform improves connectivity among the stakeholders. Companies and organizations can easily add their contact information to their target audiences and target customers. Thus, the viewers will easily find their desired organization (Leite, L. 2021a).

### 2.1.3 For families and stakeholders

Families can quickly discover schools, whether public or private, by their location, whether they provide laptops, bookshops, and many more. The Edupreneurs allow families total control over how to find the desired school. Stakeholders can search for firms and organizations by name, location, and category (Leite, L. 2021a).

## 2.2 Brainstorming of Edupreneurs

Brainstorming helps generate ideas quickly. There are several ways to create ideas for solving a problem or issue. However, the most common way is with a group of individuals. The brainstorm shows what the user can do in the application, such as searching for schools, companies, or organizations. Users can navigate various websites, such as organizations, blogs, companies, and schools. Users can register as well as use the application's services. As seen below, the Edupreneurs application brainstorm has many ideas.

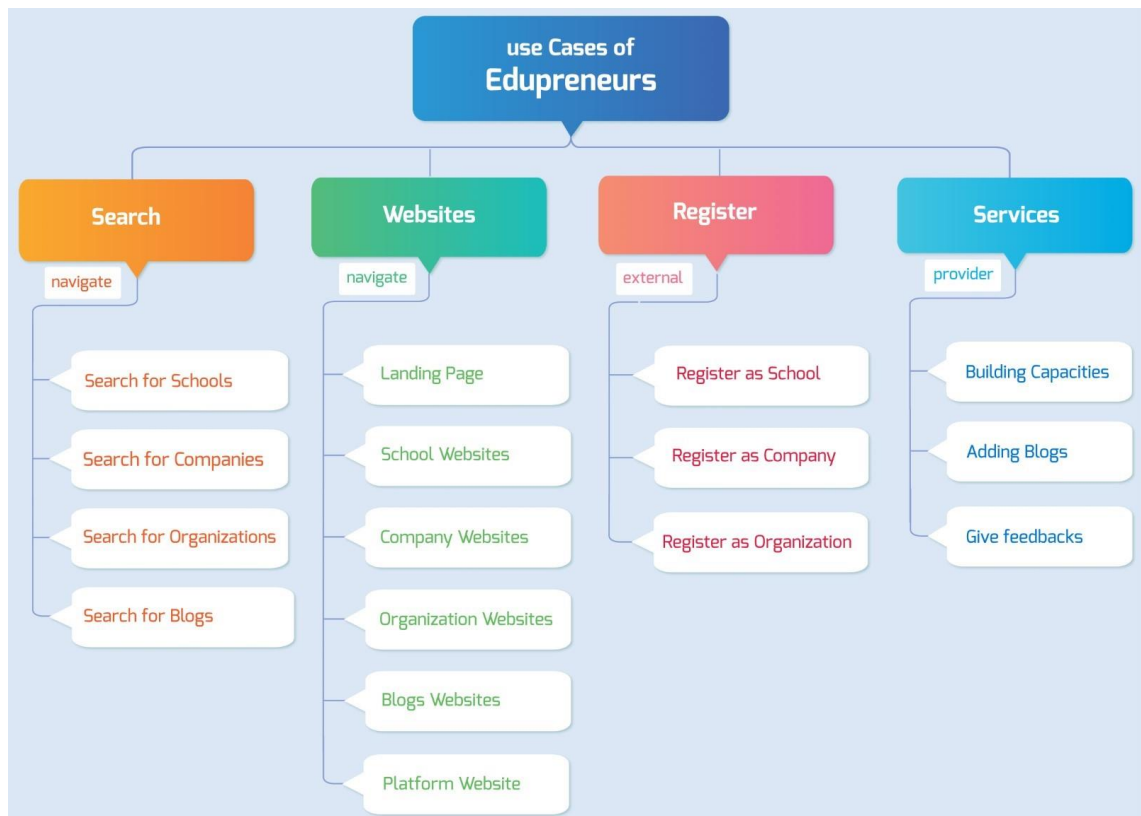


FIGURE 1. Brainstorming of Edupreneurs app

## **2.3 Strategy of Edupreneurs**

The primary strategy of Edupreneurs is as follows:

- To establish a digital strategic partnership platform in which educational and corporate institutes may gather and process data on prospective consumers and providers throughout the area, connect with key stakeholders (B2B and B2C) and promote their services and goods (Leite, L. 2021b).
- To increase digital and business competence through capacity building, education stakeholders, software developers, empowering modernizing education processes, innovating business towards sustainable economic growth, and resilience in challenging times (Leite, L. 2021b).
- To create a policy paper on developing a resilient business environment, we will combine practical experience with knowledge development, culminating in a policy document to assist Southern African governments in combining robust start-up ecosystems in sectors other than EdTech (Leite, L. 2021b).

### **2.3.1 How Edupreneurs collaborate**

Edupreneurs collaborate by creating four variables,

- B2B (Cooperation for partnership)
- B2C (Co-designing your solution)
- Business Training (M-Learning)
- Opportunities (Blog and social media)

### **2.3.2 Business training of Edupreneurs**

Business training covers marketing, product creation, network building, business management, and machine learning. The marketing section discusses content marketing and the business-to-business (B4B) paradigm (Leite, L. 2021c).



### 2.3.3 B2B vs B2C

Comparison between the B2B and B2C models (Leite, L. 2021c.).

TABLE 1. B2B vs B2C

	B2B	B2C
Meaning	Business to Business	Business to Consumer
Focus	Cooperation for partnership	Co-designing your solution
Infrastructure Suppliers	Books, Office Materials, Furniture, Internet/Internet Connectivity, LMS/EMS	Books, Office Materials, Furniture, Internet/Internet Connectivity, LMS/EMS
Service providers	Teachers and pedagogical coordinators	Teachers and pedagogical coordinators
Schools	-	Public and private schools, education centre
Submission request	Allowed	Allowed

### 3 REACT TECHNOLOGY

JavaScript's library React, developed for Facebook in 2011 to design quick and interactive user interfaces, is the most used JavaScript library globally. According to Google Trends, React dominates the library field (React Library n.d.a).

#### 3.1 React components

Components are central to all React applications, built with a range of independently isolated, reusable components. These components construct complex user interfaces known as the "root component." Each React application has at least one component that can be isolated and then combined into complicated user interfaces in implementation. Facebook has more than 10,000 components written in React (React Library n.d.b.). This method automatically updates child component properties when any element of this mother component changes and the flow goes from top to bottom; always synchronized, never duplicated.

#### 3.2 React DOM and React DOM server

The single-page application (SPA) typically uses a JavaScript bundle on the client. Even individuals who do not use JavaScript code can use server-side rendering. The React DOM server (React Library n.d.c) allows component rendering on static markers. A *React element* is a simple JavaScript object that maps into a DOM element. When a change happens to a component, virtual DOM will reference the respective element of the components, and its child component will react to it. React is distinct from vanilla JavaScript and jQuery, which use a virtual DOM.

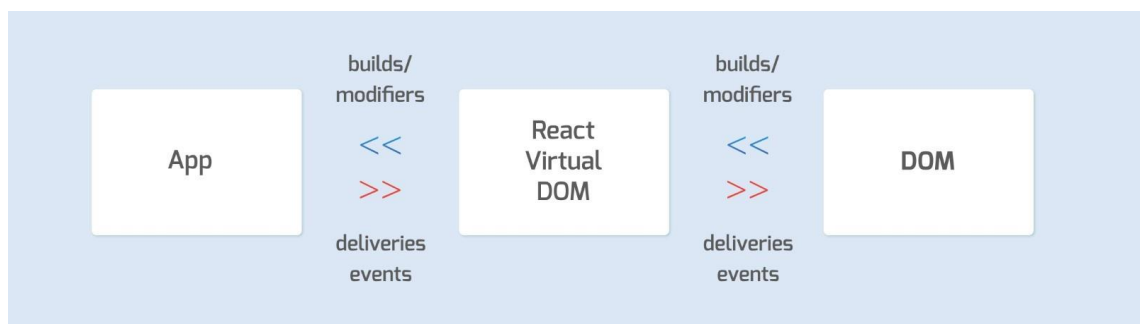


FIGURE 2. React DOM cycle

### **3.3 State vs Props**

React has the State object that lets components generate and maintain their data. React components cannot supply data with the state but can be buildable and maintainable and use properties to communicate data between React components, also referred to as Props. React Props are equivalent to a JavaScript function call passed to components via HTML attributes, unidirectional. Components can use props to obtain data from the outside world, but they can also use the state to produce and manage their data. Props move the data between components, whereas the state keeps it. Parent components can easily pass down Props to the child component. On the other hand, components can modify the state data (Bain, L. 2016).

### **3.4 React lifecycle hooks**

React offers numerous built-in methods that can be readily accessible in a class component at specific life cycle points. For React version 16.4, the life-cycle methodologies have three phases: updating, assembly, and managing errors. The updating life-cycle methods get initiated when a component's properties or status change. It starts to assemble its constructor based on a component's removal from the virtual DOM. Limits of Error handled during rendering, life-cycle methods, and constructors. The unmounting life-cycle method invokes an error while rendering a life-cycle function or constructor of a child component. The updating phase has itself four methods: constructor(), a static method that takes a derived state from the props render, and componentDidMount(). The shouldComponentUpdate() renders obtaining a snapshot before updating. The componentDidUpdate() renders obtaining a snapshot while unmounting. Before a component is unmounted and destroyed, componentWillUnmount() start invoking. The componentDidCatch() handles unexpected errors while rendering the component (React Library n.d.d).

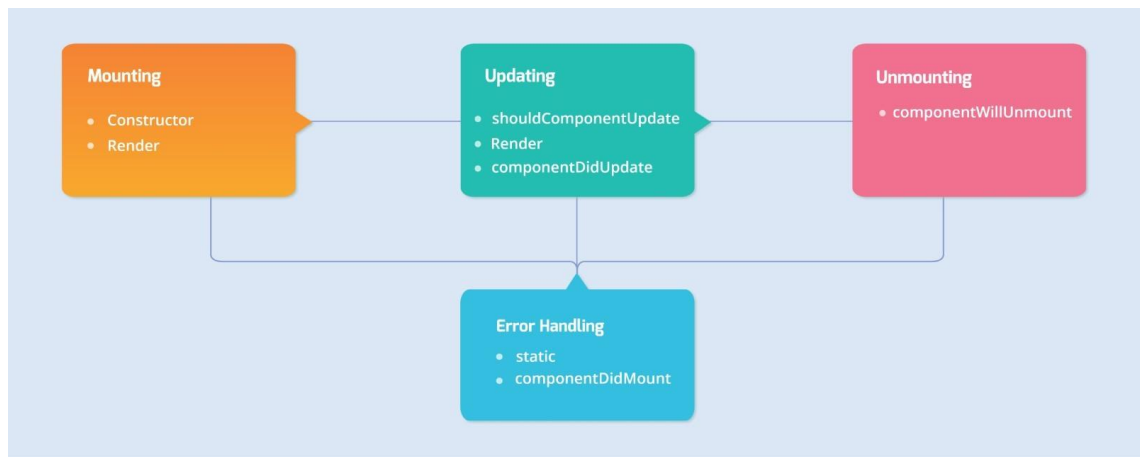


FIGURE 3. Architecture of React Life Cycle Hooks

### 3.5 Event handling

React elements have properties such as `onClick()`, `onSubmit()`, `onMouseOver()`, etc. Events are visible when a user performs a certain action based on the DOM element. It is described in-class components and functional components differently. In-class component handling event handlers need a method; unlike functional components, they need an internal or external function to handle the event. While calling the method in the class component, one must provide the 'this' keyword (React Library n.d.e).

### 3.6 Logical rendering

A React component determines which DOM elements to return based on one or more conditions in a logical render. Then, based on the status of the application. React's conditional rendering operates similarly to JavaScript's conditions do. Components reflect the current state using JavaScript operators. If the state changes by any margin, React changes the UI (User Interfaces) to match them (React Library n.d.f) accordingly.

For example, a visible state name initially set to false may subsequently be defined conditionally in the render method. If visible is true, display the button; otherwise, display something else, a simple example of logical rendering.

```

1  export default class ButtonComponent extends React.Component() {
2    constructor() {
3      super();
4      this.state = { visible: false };
5    }
6    handleClick = () => {
7      this.setState({ visible: !this.state.visible });
8    };
9    render() {
10     return (
11       <div>
12         {this.state.visible ? (
13           <Button onClick={this.handleClick}>Button</Button>
14         ) : (
15           <p>No button</p>
16         )}
17       </div>
18     );
19   }
20 }

```

FIGURE 4: Logical Rendering

### 3.7 Stateful vs Stateless component

Functional or stateless components are always re-rendered without producing an instance during the render life cycle of their parent. When used in massive quantities, that may cause performance issues. However, no optimization memorization can be performed without an instance because there is no initial render state to monitor. Functional components minimize users' vulnerabilities when they become more sophisticated, and child components become Stateful. (Huergo, F. 2019). In the picture below, Stateless is just a functional component without having any state and getting a props name message from the mother component's Stateful's state.

```

1 // stateless component
2 const Stateless = ({ message }) => <div>{message}</div>;
3 // statefull component
4 export default class Stateful extends React.Component {
5   constructor() {
6     super();
7     this.state = { message: "Hello" };
8   }
9   render() {
10    return <Stateless message={this.state.message} />;
11  }
12 }

```

FIGURE 5: Example of Stateful and Stateless component

### 3.8 React hook and its uses

React hook APIs give an alternative to creating class-based components and supply an alternate means of state administration. Functional components have life-cycle hooks that can deal with React's local state, effects, and context: `useState()`, `useEffect()`, and `useContext()`. React Hooks also expose useful hooks such as `useReducer()`, `useCallback()`, `useMemo()`, `useRef()`, and so on. Hooks refers to a class component's examples that give access to the State and lifecycle methods. The hooks offer many advantages for developers and improve how they create components (React Library n.d.g). The `useState()` hooks have the same function as the class component (`this.state`). The `"this.setState"` method is used in the class component to change the State. However, this method is set directly next to the state name in the functional component. The `useState()` hook must begin with a value. The lifecycle method `componentDidMount()` handles the user action when a component is mounted. The `useEffect()` hook, on the other hand, is invoked when the program is first mounted and performs certain specified duties for a functional component. Dependencies, like the `componentDidUpdate()` lifecycle hook in the component class, can be provided in an array to use the `useEffect()` conditionally.

```
1 import { Component, useState, useEffect } from "react";
2 // Class Component example with lifecycle hook
3 export default class ClassComponent extends Component {
4   constructor() {
5     super();
6     this.state = { message: "Hello Word" };
7   }
8   componentDidMount() {
9     this.setState({ message: "New Word" });
10  }
11  render() {
12    return <div>{this.state.message}</div>;
13  }
14 }
15 // Functional Component example with lifecycle hook
16 export const FunctionalComponent = () => {
17   const [message, setMessage] = useState("Hello World");
18   useEffect(() => {
19     setMessage("New World");
20   }, []);
21   return <div>{message}</div>;
22 };
```

FIGURE 6: Comparisons between React class component vs React Hook

### 3.9 Reusable components

In React, a reusable component is a UI element that constructs more than one UI instance in various areas of an application. For example, one may show a component in various colours in many areas of an app. However, the component's composition will remain the same regardless of the dataset used. It changes and produces a UI instance for the element. This pattern is essential for creating reactants at scale. It helps save time by guaranteeing that less code produces a faster development (Eze, P. 2019). That is a good illustration of how a component is reused and then built into another with more properties. By simply calling those individual components and supplying the props, the code can have fewer lines of code. This method refers to DRY, meaning "Do not repeat yourself."

## **4 UP AND RUNNING WITH EDUPRENEURS**

This section of the thesis deals with installing React, choosing the third-party libraries, structuring the folders and files, fetching data from REST APIs, using React Bootstrap 4, dockerizing the application, and deploying the app to the Docker registry.

### **4.1 Installation of React app**

To install an application, one must have a node installed in their environment. After installing the node, the user can either install the React application from scratch or create it all at once. An easy and more sophisticated way of doing things for beginners is to use the `npx create-react-app [app-name]` command. It creates everything behind the scenes and removes all the Webpack and Babel configuration complexity (Rascia 2018).

### **4.2 Choosing third-party libraries**

Choosing third-party libraries is very crucial. While developing the application, one needs to use a third-party JavaScript library such as Bootstrap, Axios. Developers, organizations, and communities are building and supporting many packages. The use of this third-party library contributes to the development of the application and helps achieve the aim (Kunwar, S. 2018). Dependencies used in the Edupreneurs Application are listed below.

#### **4.2.1 Axios (v0.21.1)**

Axios is an HTTP client committed to and functions in the browser and the Node.js environment. It offers one single API for XMLHttpRequests and the HTTP interface node. In addition, a polyfill for the syntax of the ES6 promise wraps the requests together. Almost any dynamic project needs some interface using RESTFUL APIs (Axios Library. n.d).



## 4.2.2 Http-Proxy-Middleware (v2.0.0)

A fetch can contain API keys as headers for development and production versions of a real-world application, and development users prefer a similar shape for that version. The proxies need context and alternatives. The context begins with the API, with which the app should communicate on both development and production servers. Then the option field can have more methods, such as `createProxyMiddleware()`. It includes `target`, `change origin`, `source`, `path rewrite`, `headers`, `onProxyReq()`, and many more (HTTP-proxy-middleware Library. n.d).

```
1  const { createProxyMiddleware } = require("http-proxy-middleware");
2  require("dotenv").config();
3
4  module.exports = function (app) {
5    const { NODE_ENV, REST_ADDRESS, API_KEY } = process.env;
6    const target = NODE_ENV === "development" ? "http://192.111.1.123:12345" :
7                  NODE_ENV === "production" ? REST_ADDRESS : "";
8    const apiKey = NODE_ENV === "development" ? "eduPre2215" :
9                  NODE_ENV === "production" ? API_KEY : "";
10   app.use(
11     createProxyMiddleware("/api/**", {
12       target,
13       secure: false,
14       changeOrigin: true,
15       pathRewrite: { "^/api": "" },
16       headers: { apiKey, origin: null },
17       onProxyReq: function (proxyReq, req, res) {
18         proxyReq.setHeader("accept-encoding", "identity");
19       },
20     })
21   );
22 };
```

FIGURE 7. HTTP-Proxy-Middleware in Edupreneurs

## 4.2.3 SWR (v0.5.6)

SWR is an HTTP invalidation method popularized by HTTP RFC 5861: stale-while-revalidation. SWR is a mechanism for initially returning cached data (stale), fetching data, and updating the data. The SWR updates components continuously and automatically with a series of data. The interface is always quick and dynamic (Liu, J. n.d). In the example below, the `useSWR()` hook takes a key string and a `getQueries()` method. The `getQueries()` method will use this key for data identification and distribution. The Axios fetcher hook handles asynchronous data

fetching for the project. The hook delivers two values, "data" and "errors," depending on the request.

```
1 import useSWR from "swr";
2 import axios from "axios";
3
4 export const getQueries = async (url) => {
5   const { data } = await axios.get(url);
6   return data;
7 };
8
9 export default function Profile() {
10  const { data, error } = useSWR("/api/user", getQueries);
11  if (error) return <div>failed to load user</div>;
12  if (!data) return <div>loading!</div>;
13  return <div>hello, {data.name}</div>;
14 }
```

FIGURE 8. SWR integration example with Axios

#### 4.2.4 React-Router-DOM (v5.2.0)

React is a library rather than a framework for creating large-scale applications. A substantial number of pages may be needed, and navigating these pages will require a third-party library. There are several options, among which React Router DOM is the most common. BrowserRouter, HashRouter, Link, Switch, Route, useHistory(), and useParams() are some of the more intriguing objects in the React Router DOM. It is best to wrap HashRouter's BrowserRouter at the top of the Root component and set the Switch at the top of App component that navigates the entire page. It should not include the Header and Footer components because they are always in the same position (React-Router-DOM Library n.d). The useHistory() hooks are convenient when dealing with complicated conditions, such as pushing and checking the location, old path, and current path. The useParams(), on the other hand, gathers the id independently while routing a page by id, allowing for helpful fetching. The Link component uses an anchor tag, and it requires a pathname and navigates according to that pathname.

#### 4.2.5 Bootstrap (v4.6.0)

Bootstrap is a large, useful, reusable collection of HTML, CSS, and JavaScript-written code parts (Bootstrap Library. n.d). It is also a borderline framework for development that helps developers construct completely responsive websites rapidly. Bootstrap saves a bunch of CSS code to write and gives more time to construct websites. To gain complete hands-on experience with the Bootstrap style in the Edupreneurs application, importing into the index.js or the app.js file.

```
1 import React from "react";
2 import App from "./App";
3 import ReactDOM from "react-dom";
4 import { HashRouter } from "react-router-dom";
5 import reportWebVitals from "./reportWebVitals";
6 import "bootstrap/dist/css/bootstrap.min.css";
7 import "./index.css";
8
9 ReactDOM.render(
10   <HashRouter>
11     <App />
12   </HashRouter>,
13   document.getElementById("root")
14 );
15
16 reportWebVitals();
```

FIGURE 9. Importing Bootstrap.

#### 4.2.6 React Bootstrap (v1.5.2)

React-Bootstrap is a top-level framework for React apps using Bootstrap underneath the hood. The components were designed without unnecessary dependencies, such as jQuery, making them usable as pure React components. React-Bootstrap has evolved and expanded, with React as one of the earliest React libraries to become an intelligent choice as a UI base (React-Bootstrap Library. n.d). The React component allows control over the component's shape and function.

### 4.2.7 React Bootstrap icons (v1.4.0)

React Bootstrap Icon is a collection of free and high-quality icons (Nathan, K. n.d). To utilize the icon, first, check the React bootstrap icon availability chart to see whether one is free or not. It can be imported with the same name and invoked as a React component, and it is possible to choose the icon's size and colour.

```
1 import { Archive } from "react-bootstrap-icons";
2
3 export const App = () => {
4   return (
5     <div>
6       <Archive color="royalblue" size={96} />
7     </div>
8   );
9 };
```

FIGURE 10. Uses of React Bootstrap Icons

### 4.3 Creating file and folder structure

Making a proper file structure gives an application a helpful management system and sustainability overall (File Structure System. n.d.). One production folder is quite helpful after it goes to production build. The public folder index.html contains the div element id of the app. The src folder is not, by default, structured. In React, there are no such rules about structured folders and files, but it is vital to do so to have more maintainability. Images, fonts are stored in the assets folder.

In contrast, the components folder contains pages and reusable functions. The config folder contains configuration files, including colours, external links, routes, and styles. The data folder contains JavaScript files with data objects in arrays. The hook folder holds all the custom hooks, and the pages folder has the main routes. The utils folder holds all utility functions, and the services folder contains all functions to fetch data.

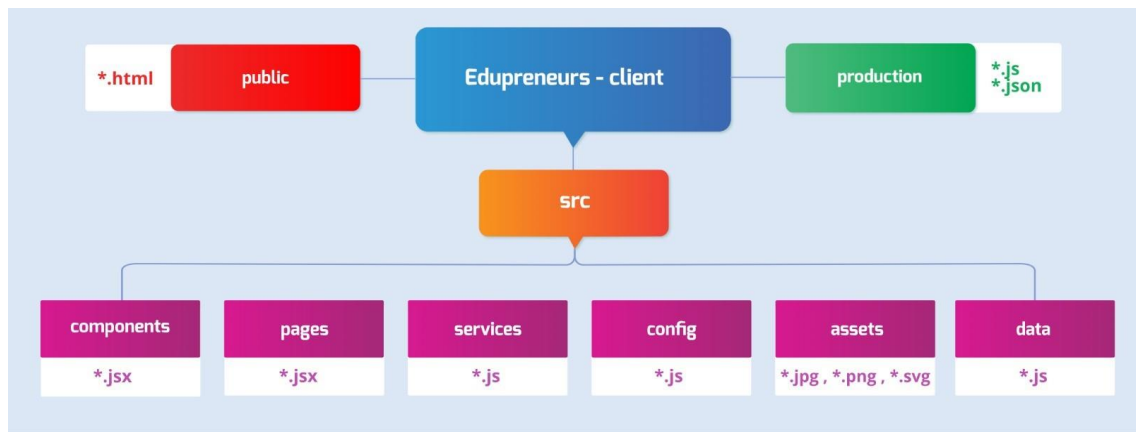


FIGURE 11. File and folder structure of the Edupreneurs app

#### 4.4 Fetching REST API via SWR

The most crucial aspect of the Edupreneur application is fetching using SWR (Liu, J. n.d). REST APIs must obtain all information on schools, companies, organizations, blogs, and alternatives. The backend of Edupreneurs connects to FormJack's Glowdom database. It is unnecessary to use the header and fetch link because the proxy is already in place. The structure of REST APIs' is listed below:

```

1 GET {{host}}/options
2 GET {{host}}/companies
3 GET {{host}}/companies/{id}
4 GET {{host}}/companies/options
5 GET {{host}}/organizations
6 GET {{host}}/organizations/{id}
7 GET {{host}}/organizations/options
8 GET {{host}}/educations
9 GET {{host}}/educations/{id}
10 GET {{host}}/educations/options
11 GET {{host}}/blogs
12 GET {{host}}/blogs/{id}
13 GET {{host}}/blogs/latest
14 GET {{host}}/blogs/options
15 GET {{host}}/attachments/{formId}/{recordId}/{attachmentId}
  
```

FIGURE 12. List of REST APIs

## 4.5 Typography and Colours choice

Typography and font choice (Font Choice. 2021) are crucial parts of any user interface. Wrong colours and typefaces give users the wrong impression (Munro, L. 2019). Through careful research, choosing these factors can improve the product's appearance. The typefaces and colours used in the Edupreneurs app are listed below:



FIGURE 13. Typography and Color selection

## 4.6 Iconography of Edupreneurs

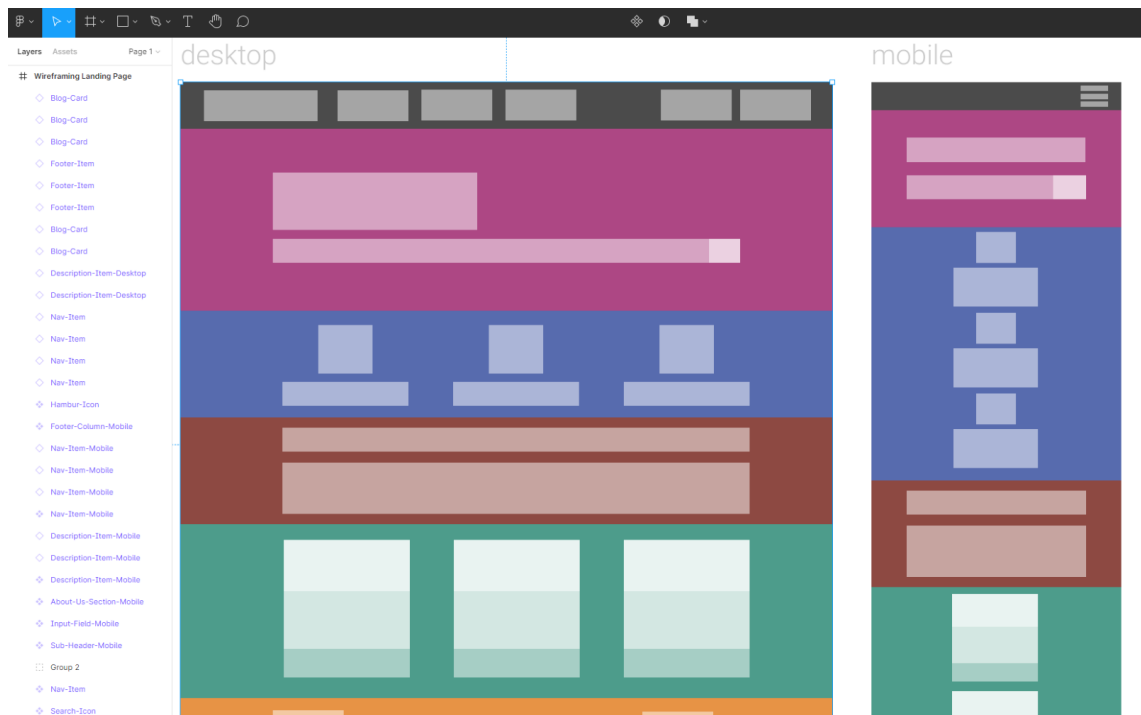
It is easy to draw consumers into the content of a website or mobile app by using icons. Using icons, one can rapidly summarize the content. It is easy to convey the main idea of a product or service using icons. Suppose one wants to draw attention to paragraphs and other information blocks instead of traditional bullet points. In that case, one may utilize visually appealing symbols. Edupreneurs use the following icons:



FIGURE 14. Icons used in the Edupreneurs app

#### 4.7 Wireframing of Landing Page

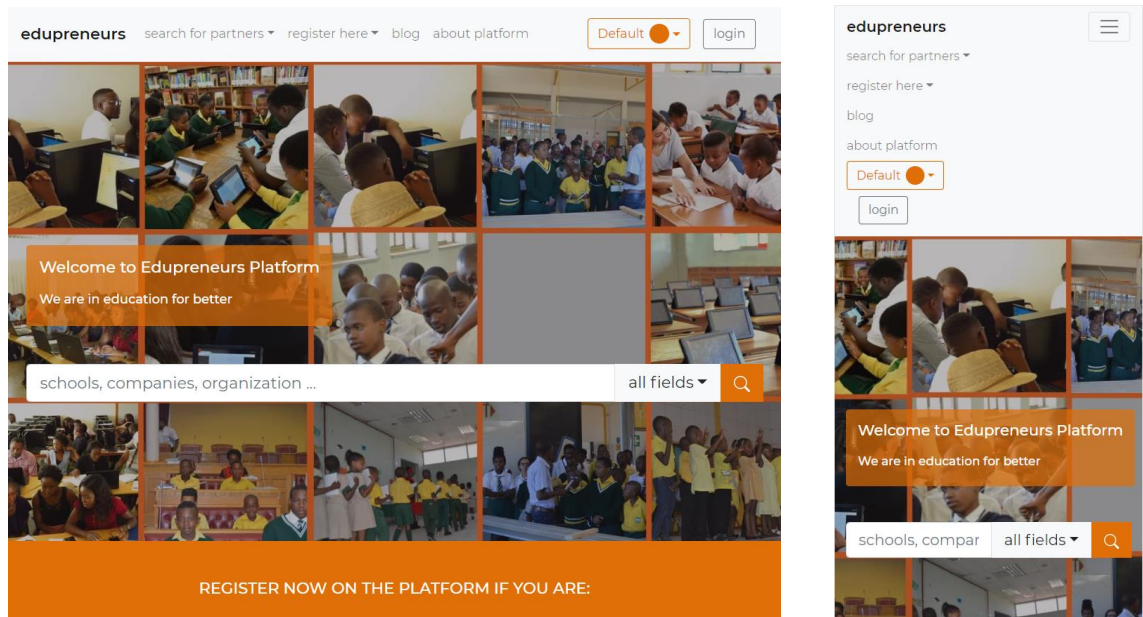
By using a wireframe, one can use basic shapes and text components to stand for page structure, content, and navigation. As an alternative, one can create wireframes by hand or by using a tool such as Adobe XD or Figma. Here is an example of the Edupreneurs wireframing for the landing page.



PICTURE 1. Wireframing of the Edupreneurs application

## 4.8 Desktop vs Mobile View in action

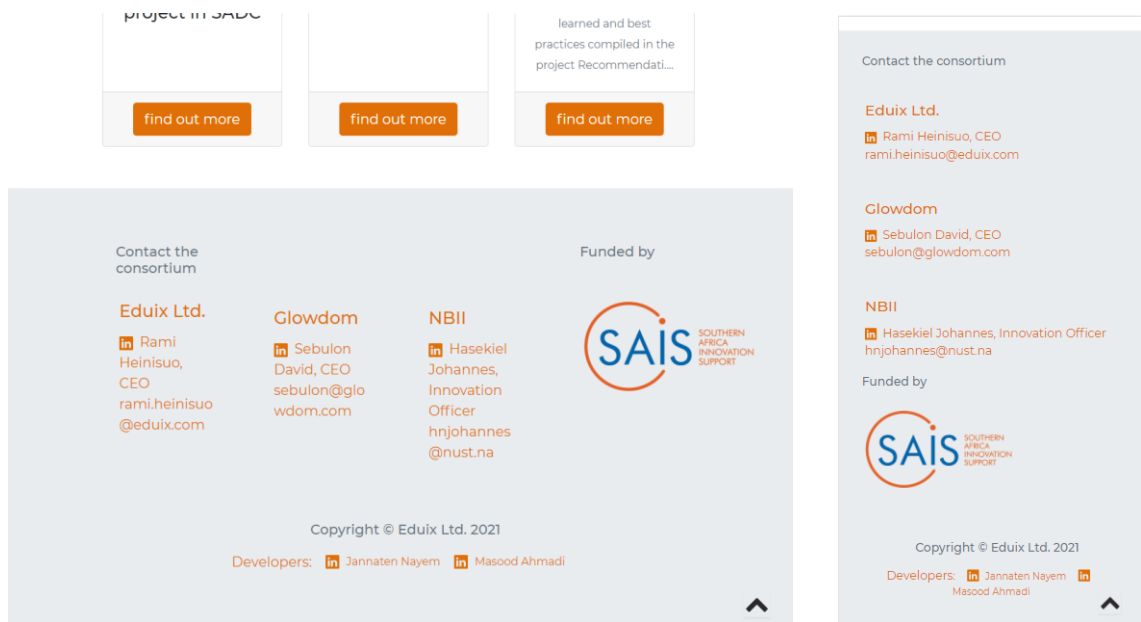
The navigation bar is the most apparent difference between the desktop and mobile versions. Anything below 993px is considered a tablet or mobile version. The navbar items are hidden in the mobile version until the user clicks the hamburger icon.



PICTURE 2. Desktop and Mobile — Landing Page

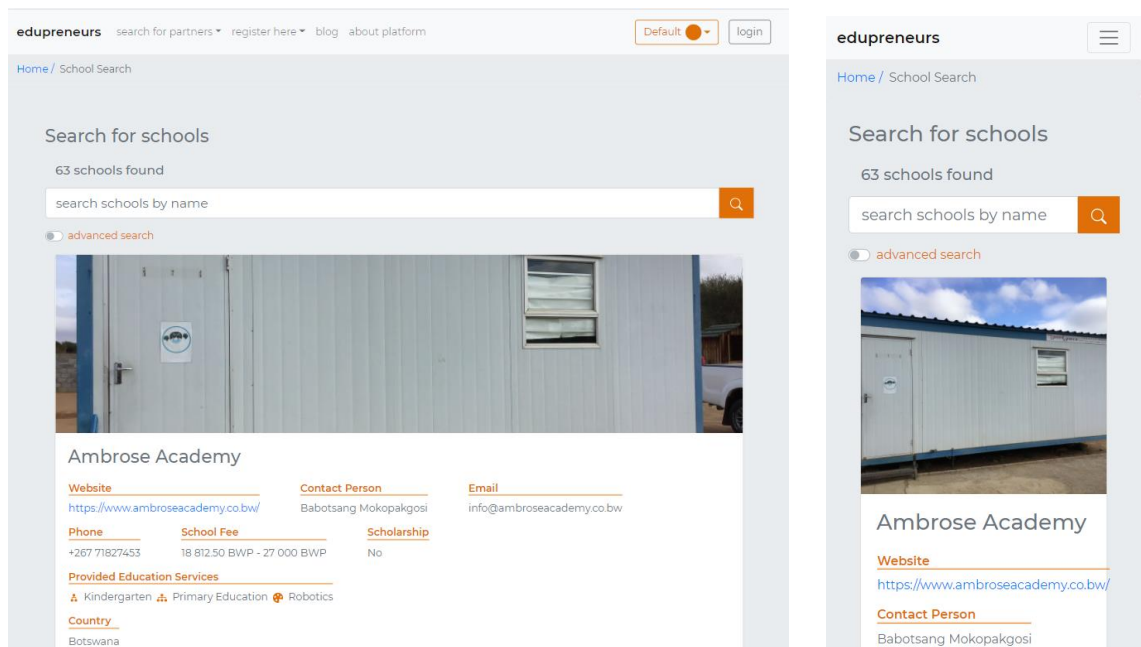
Because the width of the desktop version is too large, the footer elements list row-wise, while the flex-direction has the property of a row. The footer elements are listed column-wise in the mobile version. The flex-direction property is selected column-wise and becomes more responsive as a result.





PICTURE 3. Desktop and Mobile — Footer Page

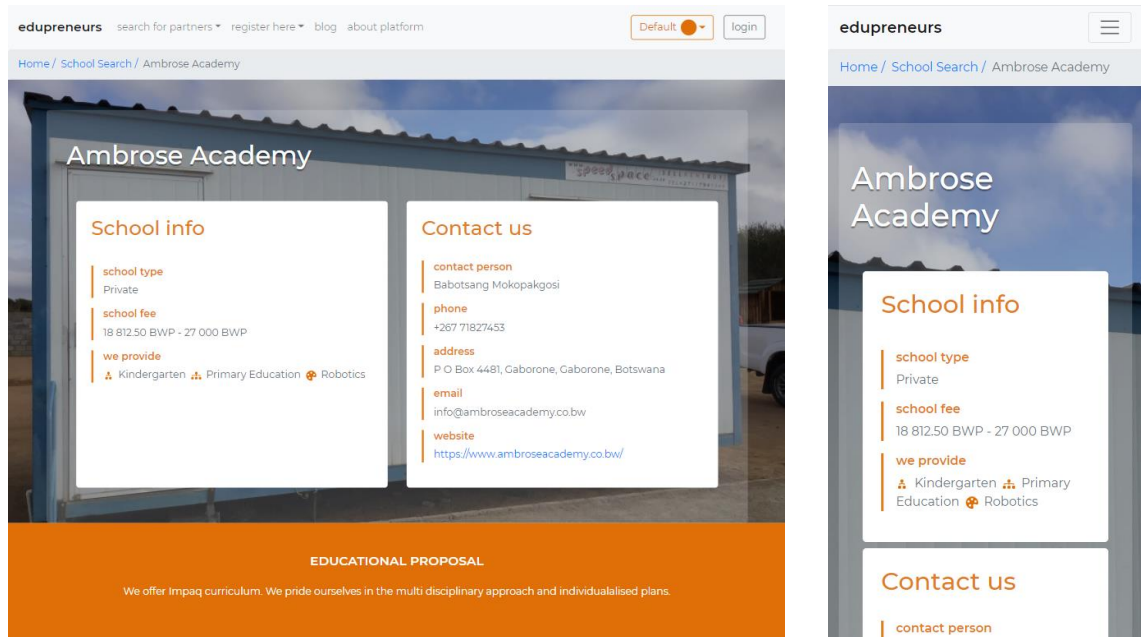
The most visible part of the school page is the image, and the cover is the image background property's value. As a result, regardless of how small or large the image is, it will always cover its entire width along with the Card element and provide a pleasant visual experience.



PICTURE 4. Desktop and Mobile — Education Search Page

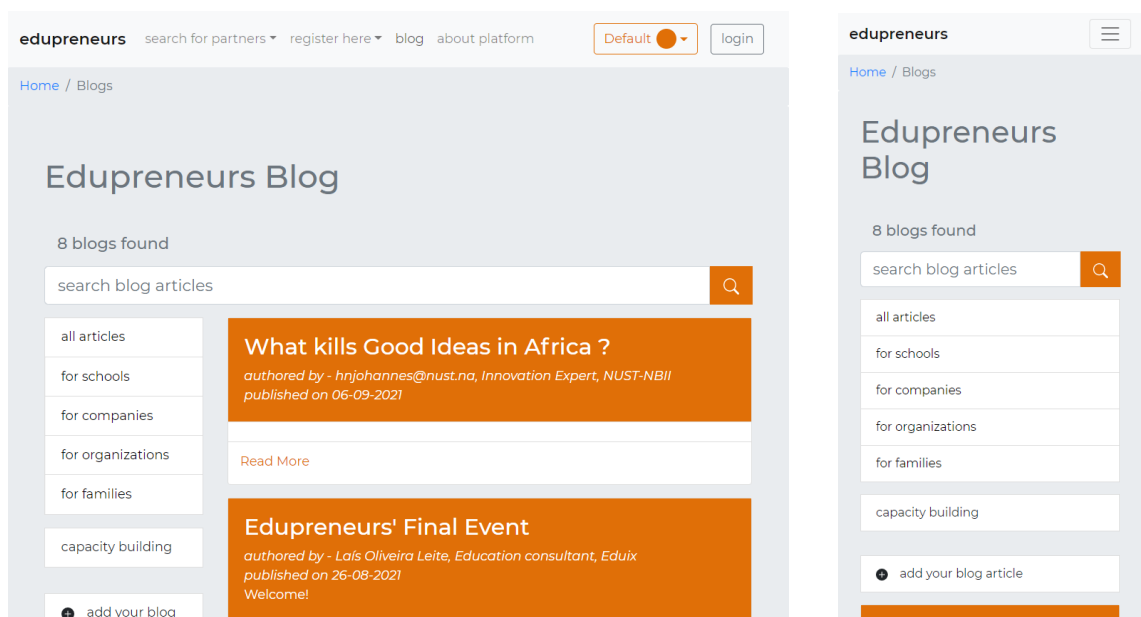
The main observable thing on the school website page is the background image behind the two-card elements. The background of this image set covers, which

means both width and height are proportionate to its parent. The moment the desktop version breaks, the card's flex-direction breaks into the row-to-column to give the user a better UI experience.



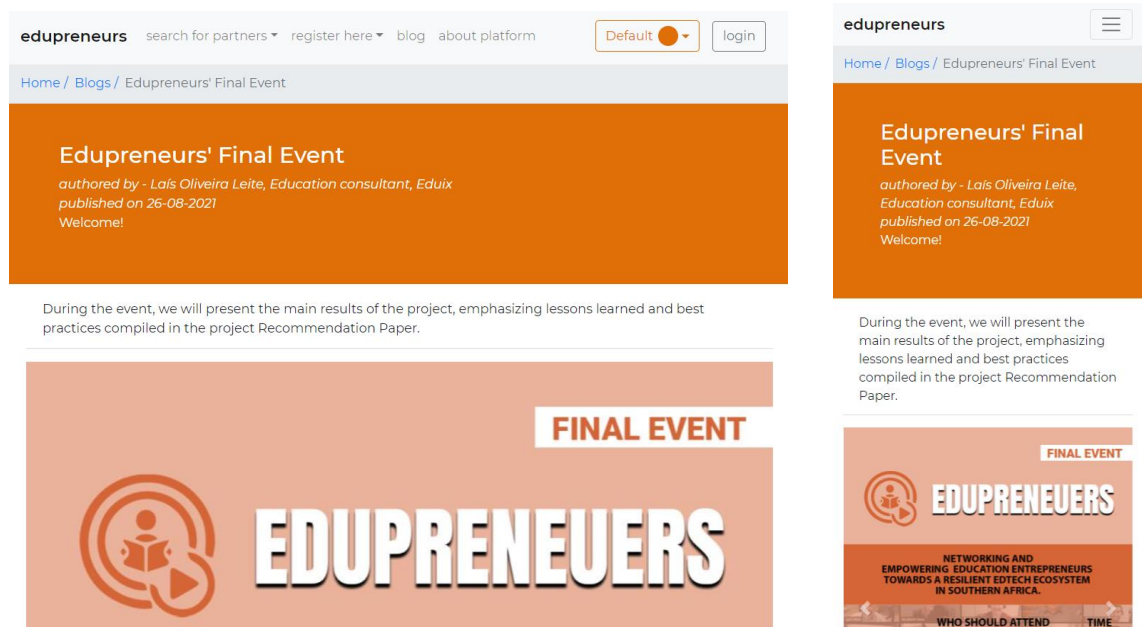
PICTURE 5. Desktop and Mobile — Education Website Page

The blog search page is visible in a folded fashion. The blog categories and blog descriptions are listed row-wise in the desktop version. In contrast, they list in a column-wise direction in the mobile version. The categories appear first, followed by the blog results.



PICTURE 6. Desktop and Mobile — Blog Search Page

The desktop and mobile versions of a blog website page almost look the same regarding responsiveness. Flex is programmed to wrap, and it is not changing its flow but stretching its components to provide a better UI experience.



PICTURE 7. Desktop and Mobile — Blog Website Page

#### 4.9 Responsive App by using Bootstrap and React Bootstrap

It is crucial to study the documentation for Bootstrap and React-Bootstrap to make the Edupreneurs application responsive. Responsiveness refers to how the app will appear on various platforms, such as desktops, tablets, and mobile phones. Before Bootstrap, responsiveness was handled simply by using CSS properties such as media-query, flexbox, grid (Flexbox and Grid System. n.d.), and jQuery JavaScript actions. However, Bootstrap hides all the complexity and gives total flexibility to get the same style and action by using some class names and methods. The CSS and JavaScript actions will be available after installing React-Bootstrap (Vijayvargiya, S. n.d.) in the working directory. One should call the CSS styles in the Root component after installing Bootstrap and React-Bootstrap using node dependencies. The root app file imports the styles files, either in app.js or index.js, by adding the bootstrap.min.css file from the node module's Bootstrap's 'dist/css' directory to the top. The Bootstrap library include predefined

class names (Bootstrap Cheat-Sheet, n.d.). The class name used in the Edupre-neurs app is listed below:

- text-muted, text-primary, text-secondary: used for coloring texts
- text-center, text-left: used for aligning text
- ml-[a], mr-[b], mb-[c], mt-[d]: used for adding margins in a template, particularly on left, right, top and bottom
- pl-[a], pr-[b], pb-[c], pt-[d]: used for adding paddings to a template, particularly on left, right, top and bottom
- h1, h2, h3, h4, h5, h6: used for giving the text a specific text font size and font-weight based on the header call. h1 has the immense font size and font weight, and h6 has the smallest font size and weight
- row, justify-content-center: used for centralizing a template and justifying the content in a row and taking the content in the centre
- pagination, d-flex, flex-wrap: pagination class used for converting a list group into pagination, d-flex make the DOM element display in flex mode and flex-wrap mode. If the display gets shorter, it will wrap the element
- page-item, page-link, shadow-none: page-item used for listing elements to act as pagination items, where page-link makes the element clickable as a button, and shadow-none removes all sorts of shadows from the element
- d-block, w-100, h-100: used for the template to display as a block. The template width becomes 100 rem and the height 100 rem
- align-item-start: used for aligning items in the flex-start position
- border-left: used for creating a border in a template on the left side
- font-weight-lighter: used for giving a font-weight much lighter than usual

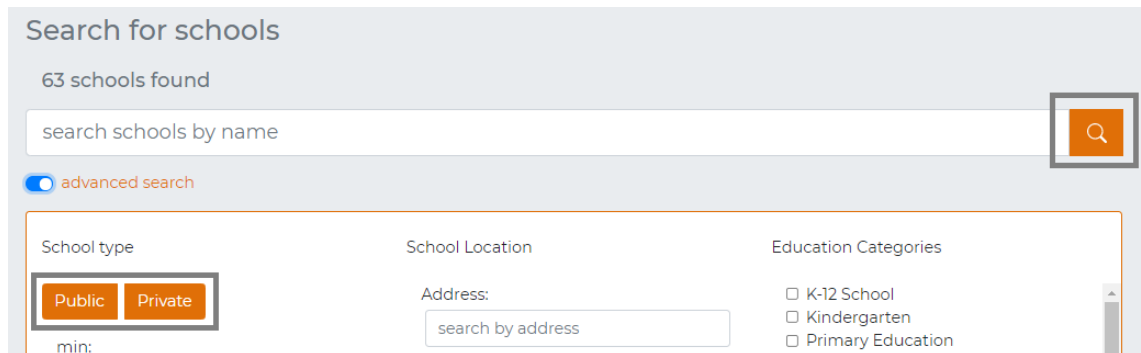
The React-Bootstrap components used in the app are listed below:

**Breadcrumb:** It displays the current page's location inside a navigational hierarchy that inserts separators automatically through CSS. The active prop should be added to the active breadcrumb and do not use the active and href properties simultaneously.



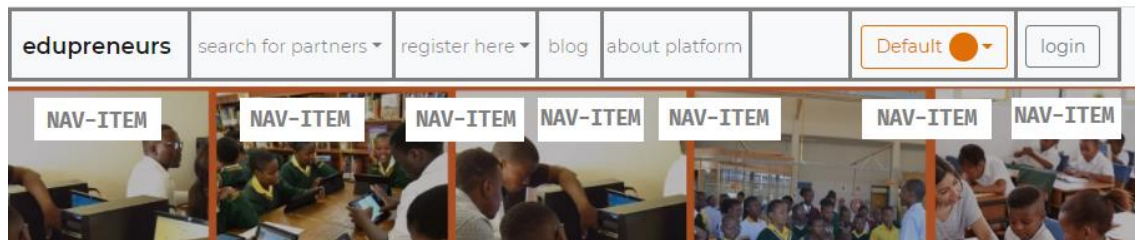
PICTURE 8. Breadcrumb example

**Button:** It is a responsive component with predefined styles such as outline, rounded, solid, and methods such as `onClick()` and `onChange()`.



PICTURE 9. Buttons example

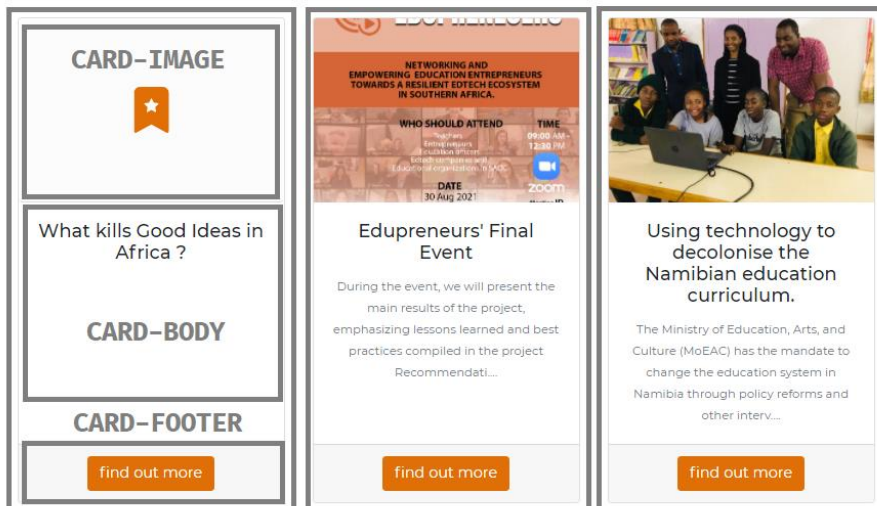
**Nav, NavBar, and NavDropdown:** Navigation bootstrap components share a general navigation component and styles. The Nav fundamental component features a flexbox and supplies a robust base for all kinds of navigation components. The navigation header is solid and responsive, and it includes branding support, browsing support, and more. The NavDropdown component holds certain parts of the list items named "Nav-Items."



PICTURE 10. NavBar example

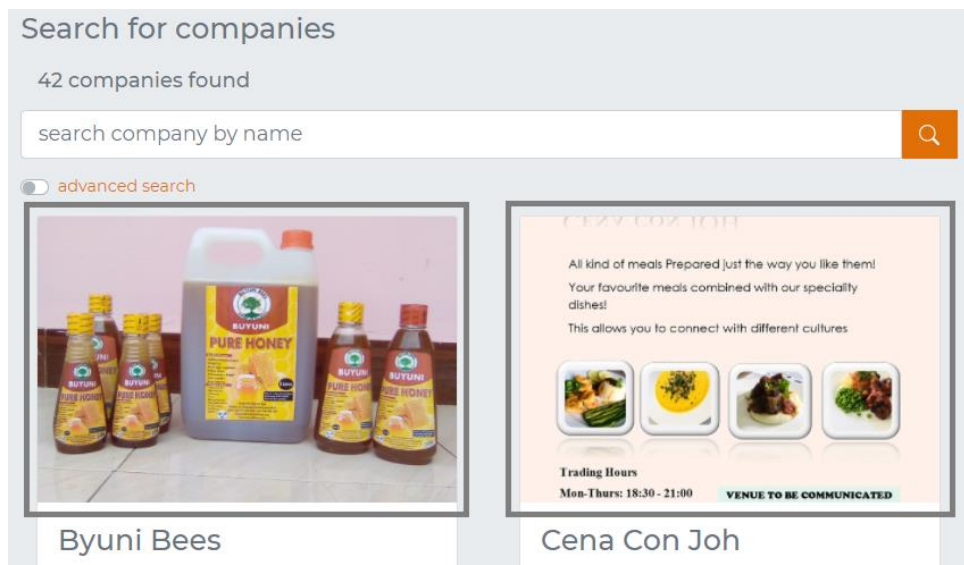
**Card:** The card component is constructed with minor possible markings and designs but supplies plenty of control and personalisation. They are made of flexbox and enable easy alignment and a good blend of other Bootstrap components. They have no default margin, and therefore, they utilise space tools as necessary.

### Selected Content



PICTURE 11. Cards example

**Image:** Bootstrap images are made responsive by the `.img-fluid` class name. The maximum width is 100%, and the height is auto for the picture so that the parent element scales.



PICTURE 12. Image example

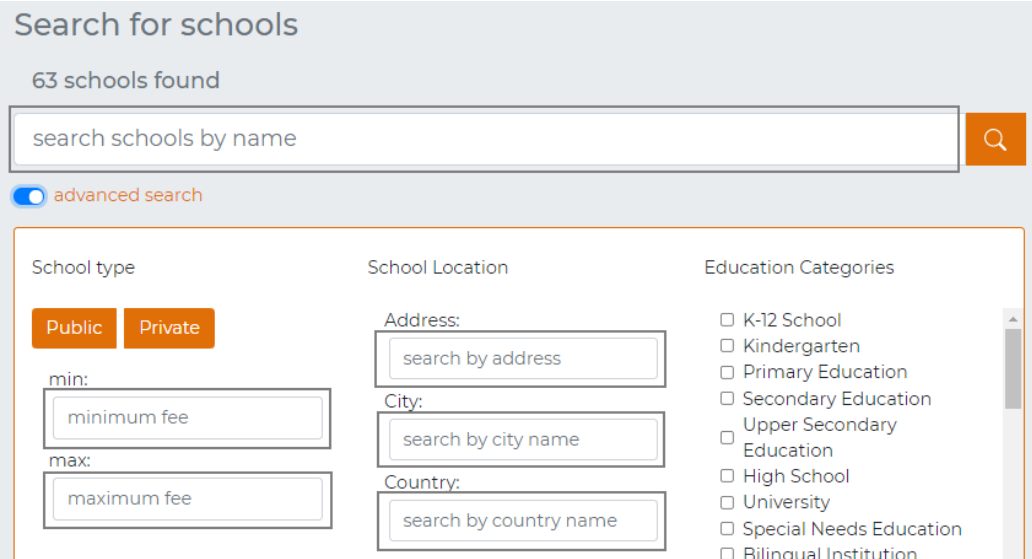
**Jumbotron:** This component is a large box to give exceptional material or information. The Jumbotron is present as an initially grey cube with rounded corners. However, the components are customizable by changing the font size of the text. Almost any legal HTML element or class is placeable on a jumbotron.

**Container:** The Container is the most fundamental Bootstrap layout element is necessary to use in the default grid system for content holding, padding, and centring. Most Container's do not require nested DOM elements, yet they can be nested.

**Col:** Bootstrap has a grid system that holds 12 grids. To use the Col component, one must define how many grids one should use. It has xs, sm, md, lg, and xl sizes where the grid can be defined accordingly. These properties are the short-hand for media queries, flex boxes, and grid layout.

**Row:** A Row component holds a card element, Col element, or other DOM element and applies where the grid is in use.

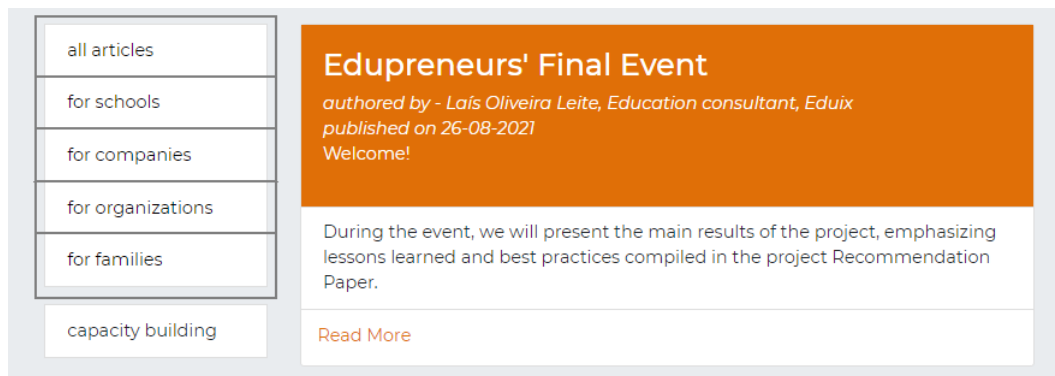
**FormControl and FormGroup:** FormControl is a user interface control that functions as the user's point of contact with the server. Interactions differ depending on the type of control. The FormGroup collects each child's data FormControl into a single object, with the control name as the key. It determines its status by subtracting the status values of its offspring.



The image shows a web form titled "Search for schools" with a light blue header. Below the title, it says "63 schools found". The main search area contains a text input field with the placeholder "search schools by name" and an orange search button with a magnifying glass icon. Below this is a toggle switch for "advanced search" which is currently turned on. The advanced search section is enclosed in a rounded rectangle and is divided into three columns: "School type", "School Location", and "Education Categories". Under "School type", there are two orange buttons: "Public" and "Private". Below these are two input fields for "min:" (with "minimum fee" as a placeholder) and "max:" (with "maximum fee" as a placeholder). Under "School Location", there are three input fields: "Address:" (with "search by address" as a placeholder), "City:" (with "search by city name" as a placeholder), and "Country:" (with "search by country name" as a placeholder). Under "Education Categories", there is a list of checkboxes with labels: "K-12 School", "Kindergarten", "Primary Education", "Secondary Education", "Upper Secondary Education", "High School", "University", "Special Needs Education", and "Bilingual Institution".

PICTURE 13. Form and FormControl example

**ListGroup and ListGroupItem:** A ListGroup component holds "li" HTML items inside it and uses list buttons and many more.



PICTURE 14. ListGroup and ListGroupItem examples

**Spinner:** The Spinner component can display the loading status. Bootstrap offers three animation choices to spin, where the Spinner component must have animation style props. It can also take custom CSS classes for further modification.

**InputGroup:** The input group hides most of the complexity of HTML, and it requires an initial value and a placeholder. It has an onChange() method that triggers the changes if the value has changed.



PICTURE 15. Input group example

**ToggleButton and ToggleButtonGroup:** Toggle buttons and groups include various buttons, such as radios and checkboxes. It hides all the complexity of regular vanilla JavaScript.

**Badge:** Badges help bring fresh or unread items to gain attention. Adding a span class name badge to links, navigation elements, and other elements is all that's needed to use a Badge component.

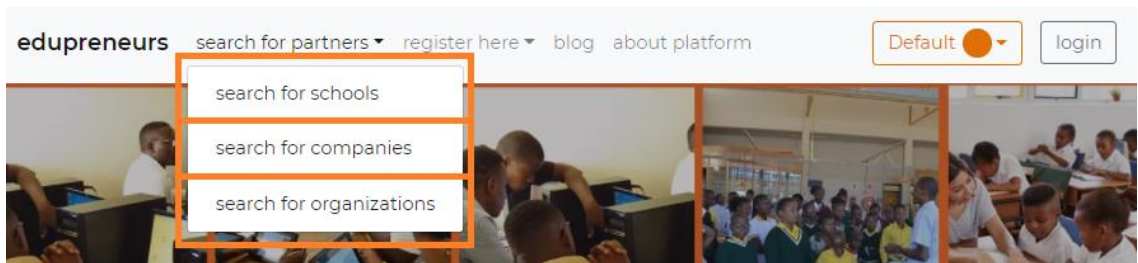
**Carousel:** The Carousel is an image slider using JavaScript to display a succession of items that use images, text, or a custom markup sequence. There is also support for previous/following buttons and indicators.





PICTURE 16. Carousel example

**Dropdown and Dropdown-Button:** Dropdowns are toggleable overlays showing link lists and other information. The JavaScript plugin included in Bootstrap makes it interactive, clickable instead of hovered over.



PICTURE 17. Dropdown and Dropdown button example

## 5 FUTURE IMPROVEMENTS

As the app grows, the app's issues keep on rising. It is essential to tackle the problem to have a durable application in the long run.

### 5.1 Solving the State Management issue

In general, the more the program gets compounded, the more the code gets redundant. State manipulation becomes much more challenging. Even if the code is bug-free, it will lead to the code in a manner that will be difficult to maintain. The complexities in the Edupreneurs application regarding state management (Dodds, k. 2020) demonstrates below:

```
1 export default function SchoolSearch({ searchField, setSearchField }) {
2   const { home } = routes;
3   const labelOne = "Public";
4   const labelTwo = "Private";
5   const history = useHistory();
6   const { primary } = useContext(ColorContext);
7   const { schoolsUrl, schoolCategoriesUrl } = endPoints;
8
9   const [pageSize] = useState(4);
10  const [currentPage, setCurrentPage] = useState(1);
11  const [searchMinValue, setSearchMinValue] = useState("");
12  const [searchMaxValue, setSearchMaxValue] = useState("");
13  const [value, setValue] = useState([labelOne, labelTwo]);
14  const [searchCityField, setSearchCityField] = useState("");
15  const [selectedCatagories, setSelectedCatagories] = useState([]);
16  const [searchAddressField, setSearchAddressField] = useState("");
17  const [searchCountryField, setSearchCountryField] = useState("");
18  const [toggleAdvancedSearch, setToggleAdvancedSearch] = useState(false);
19
20  const { data: schools, error } = useSWR(schoolsUrl, getQueries);
21  const { data: educationCatagory } = useSWR(schoolCategoriesUrl, getQueries);
22  //////////////////////////////////////
```

FIGURE 15. State Management issues.

Other components can also have the similar issues. There are tools to handle these problems on the market, such as Redux, Redux toolkits, Saga, and Apollo. Nevertheless, in a head-to-head challenge, the Context API has come to the point of fixing the same issue. The best thing about using the Context API is that one does not need to install any third-party libraries because it comes with React.

## 5.2 Implementation of Context API

Context folder's files have names according to the component's contents. These files have states and methods (Context API n.d.) that are easily accessible to the child components.

```
1 import useSWR from "swr";
2 import { endPoints } from "../././config";
3 import { getQueries } from "../././services";
4 import { useState, createContext } from "react";
5 // Creating context
6 export const BlogsContext = createContext();
7 // Providing context
8 export const BlogsContextProvider = ({ children }) => {
9   const { blogUrl } = endPoints;
10  const [pageSize] = useState(3);
11  const [currentPage, setCurrentPage] = useState(1);
12  const { data: allBlogs, error } = useSWR(blogUrl, getQueries);
13  const onPreviousPage = () => setCurrentPage(currentPage - 1);
14  const onNextPage = () => setCurrentPage(currentPage + 1);
15  return (
16    <BlogsContext.Provider
17      value={{
18        onPreviousPage,
19        currentPage,
20        onNextPage,
21        allBlogs,
22        pageSize,
23        error,
24      }}
25    >
26      {children}
27    </BlogsContext.Provider>
28  );
29  };
```

FIGURE 16. Context file holding State, Methods, and Values

A context file can contain all the methods used for that file and methods used by another component, such as a child.

After defining all the methods and data values, it is easy to return them. The mother component has these values as a global property. *Context APIs* are known as higher-level components that need states and methods.

```
1 //////////////////////////////////////////////////
2 { /* Blogs Routes */
3 <Route
4   path={blogs} render={(props) => (
5     <BlogsContextProvider>
6       <Blogs {... props} />
7     </BlogsContextProvider>
8   )
9 }
10 />
11 //////////////////////////////////////////////////
```

FIGURE 17. Supplying Context API's

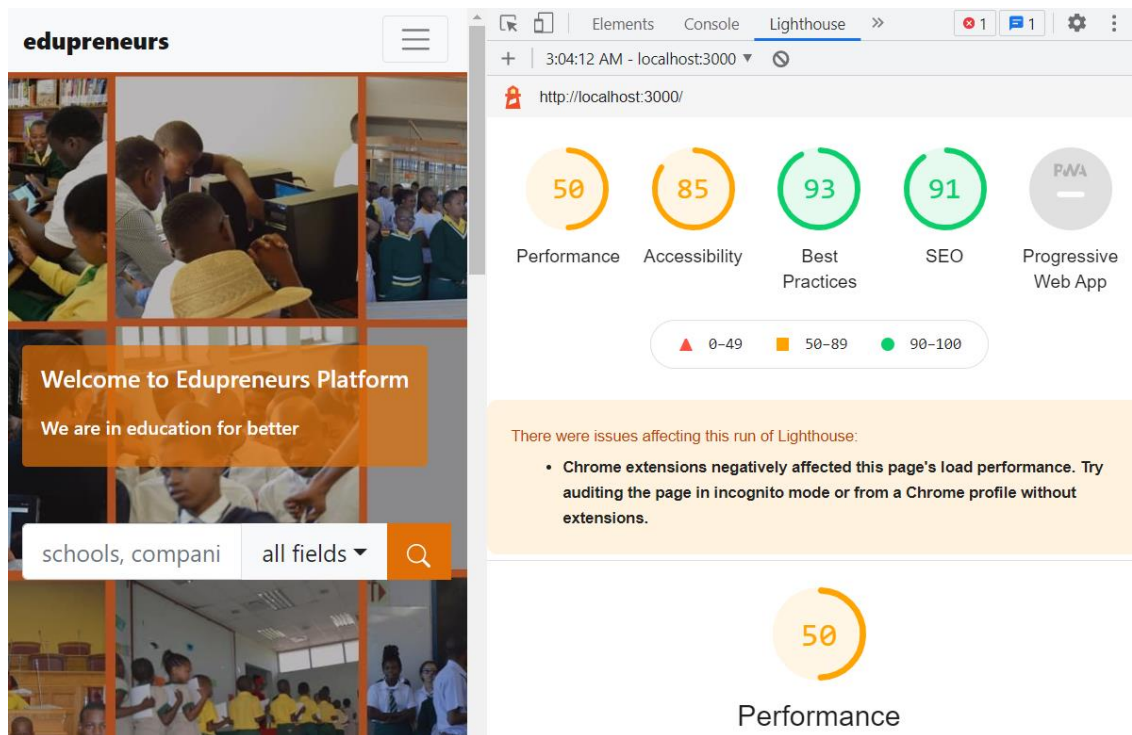
The component can now use hook calls, `useContext()`, and import the context file to extract the data.

```
1 import { useContext } from "react";
2 import { BlogsContext } from "../contexts/";
3
4 const Blogs = () => {
5   const { error, pageSize, allBlogs } = useContext(BlogsContext);
6   const { currentPage, selectedBlog } = useContext(BlogsContext);
7   const { searchInputValue, handleSearch } = useContext(BlogsContext);
8   const { handlePageChange, handleBlogSelect } = useContext(BlogsContext);
9   const { toHomePage, onPreviousPage, onNextPage } = useContext(BlogsContext);
10  //////////////////////////////////////////////////
```

FIGURE 18. Getting all the Values in the wrapped component

### 5.3 Testing performance by using Lighthouse

Lighthouse is a free, open-source tool for website audits (Lighthouse extension n.d.). Google has built a tool to analyse the performance of advanced web applications, accessibility, best practices, and SEO components. The Lighthouse app incorporates other performance analytics tools from Google, like the Page Speed Insights analysis and browser-based audits, through the development tools of the Chrome browser. The Lighthouse application gives a solid website report after performing a test.



PICTURE 18. Lighthouse report after testing

Lighthouse suggestions are listed below:

- Reduce unused JavaScript
- Preload Largest Contentful Paint image
- Ensure text remains visible during Webfont load
- Image elements do not have explicit width and height
- Serve static assets with an efficient cache policy
- Buttons do not have an accessible name
- Image elements do not have [alt] attributes
- Background and foreground colors do not have a sufficient contrast ratio
- Browser errors were logged to the console
- The web app manifest or service worker do not meet the install ability requirements
- Does not register a service worker that controls page and start\_url
- Manifest doesn't have a maskable icon

Each error shows where exactly and why the performance is low.

## 5.4 Improving application performance

After fixing all the errors and warnings, performance improved significantly. Solutions to errors are listed below.

**Image elements do not have explicit width and height:** One may see the report to fix the picture size in Google Page Speed Insights when testing a website to determine if it meets the requirements for core web vitals. There is a typical issue with dynamic apps that has to do with the width and height. As a work-around, make sure that both properties' width and height are specified, not provided in the faulty code. The website's performance improves significantly by applying this. Inline style or class properties can fix this problem if the app contains bootstrap and external CSS.

faulty - code	Correct - Lighthouse Method
<pre data-bbox="347 1003 608 1211">&lt;Image   alt="image"   variant="top"   src={NAMIBIA_LOGO}   style={{     width: "13rem",   }} /&gt;</pre>	<pre data-bbox="914 1003 1174 1189">&lt;Image   alt="NAMIBIA"   variant="top"   src={NAMIBIA_LOGO}   width={208}   height={97} /&gt;</pre>
<pre data-bbox="347 1256 852 1413">&lt;img   alt="attachment 5"   className="d-block w-100"   src={sourceImageUrl(imageFour)}   style={defaultSize ? {} : carouselImageHeight} /&gt;</pre>	<pre data-bbox="914 1256 1422 1413">&lt;img   alt="attachment 5"   className="d-block w-100 h-100"   src={sourceImageUrl(imageFour)}   style={defaultSize ? {} : carouselImageHeight} /&gt;</pre>

FIGURE 19. Image elements do not have explicit width and height

**Failing Elements:** Inconsistency in the code base is the cause of this problem. Using class names is essential when using Bootstrap for styling. The h3 tag was misused to apply the h3 effects, which is not a good approach. The failing elements problem has a solution using the h3 class name in a paragraph element.

faulty - code	Correct - Lighthouse Method
<pre> &lt;Nav className="flex-column"&gt;   &lt;Nav.Item className="text-muted ml-3"&gt;     &lt;h3 style={{ fontSize: "1rem" }}&gt;       Funded by     &lt;/h3&gt;   &lt;/Nav.Item&gt;{" "} &lt;/Nav&gt; </pre>	<pre> &lt;Nav className="flex-column"&gt;   &lt;Nav.Item className="text-muted ml-3"&gt;     &lt;p className="h3"&gt;Funded by&lt;/p&gt;   &lt;/Nav.Item&gt;{" "} &lt;/Nav&gt; </pre>

FIGURE 20. Failing element

**Lists do not contain only <li> elements and script supporting elements:** A BreadCrumbItem stands in for an unordered list missing its primary list items. It may not appear to be a problem from a visual perspective, as CSS is used to support the page. To fix this, wrapping the list outside the array containing the data, removing the problem.

faulty - code	Correct - Lighthouse Method
<pre> {breadCrumbItem.map((item, index) =&gt; (   &lt;span     className={       index === breadCrumbItem.length - 1         ? "text-muted"         : "text-primary mr-2"     }     key={item.id}     onClick={() =&gt; history.push(item.path)}     style={       index === breadCrumbItem.length - 1         ? fontSizeBasic         : fontSizeBasicWithointer     }   &gt;     {item.label}     {index === breadCrumbItem.length - 1       ? "" : "/"     }   &lt;/span&gt; )}} </pre>	<pre> &lt;li style={{ listStyleType: "none" }}&gt;   {breadCrumbItem.map((item, index) =&gt; (     &lt;span       className={         index === breadCrumbItem.length - 1           ? "text-muted"           : "text-primary mr-2"         }       key={item.id}       onClick={() =&gt; history.push(item.path)}       style={         index === breadCrumbItem.length - 1           ? fontSizeBasic           : fontSizeBasicWithointer         }     &gt;       {item.label}       {index === breadCrumbItem.length - 1         ? "" : "/"       }     &lt;/span&gt;   ))} &lt;/li&gt; </pre>

FIGURE 21. The list does not contain only the li element

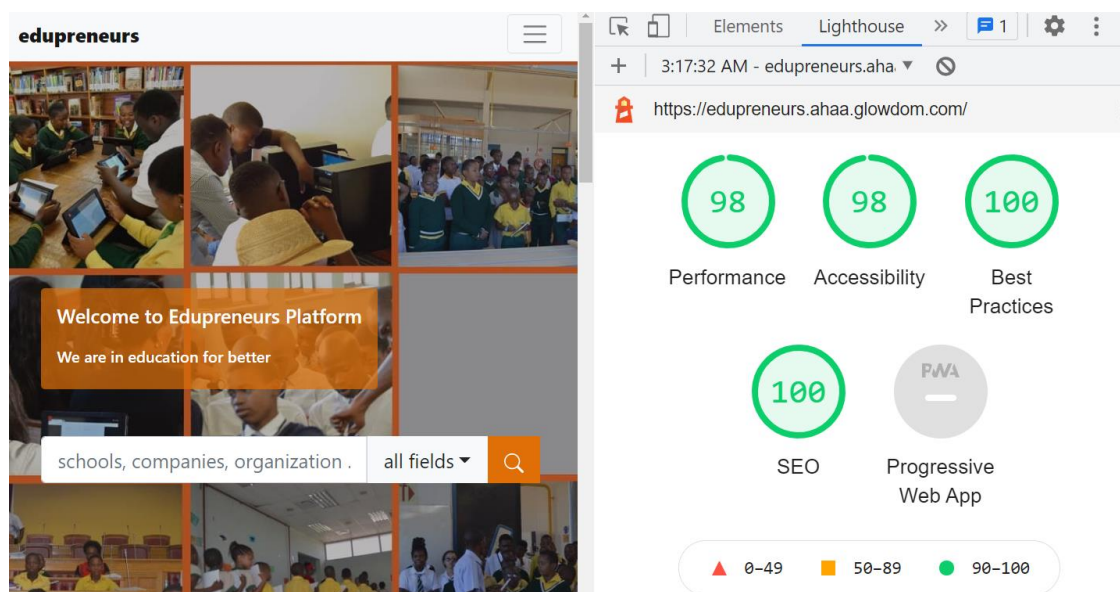
**[aria-\*] attributes do not match their roles:** Accessibility information is needed by those using screen readers or other assistive technology to understand how one's web page's controls behave and what they do. Like buttons and radio groups, many other HTML controls already have this information built into them, as do many other HTML controls. ARIA responsibilities and attributes are required for custom controls that one develops. The aria-roles specific subset of

ARIA's supports\* characteristics. When a role does not support an attribute, applying it to the role will not cause it to fail.

faulty - code	Correct - Lighthouse Method
<pre> &lt;InputGroup.Prepend&gt;   &lt;Button variant=""     style={outLineNoneWithBgPrimary}&gt;     &lt;Search style={colorBright} /&gt;   &lt;/Button&gt; &lt;/InputGroup.Prepend&gt; </pre>	<pre> &lt;InputGroup.Prepend&gt;   &lt;Button     variant=""     aria-label="search"     style={outLineNoneWithBgPrimary}&gt;     &lt;Search style={colorBright} /&gt;   &lt;/Button&gt; &lt;/InputGroup.Prepend&gt; </pre>

FIGURE 22. [aria-\*] attributes do not match their roles

After deploying the website to the production version and testing, it gives much better points in Lighthouse.



PICTURE 19. Improved performance



## 6 DISCUSSION

Edupreneurs is one of the most powerful platforms on the Southern African continent that uses EdTech infrastructure. The issues in these regions are:

- Insufficient data about education providers
- Lack of educational institute quality standards
- Difficult for parents and students to find the best providers
- Networking with different providers is hard everywhere, but seminars and good connections make it a bit easier in Europe.

The recommendation is for the Southern African area to begin adopting applications, taking full advantage of services, and building connectivity and networking.

## REFERENCES

Axios Library. n.d. Promised based HTTP client for browser and node.js. Read 27.08.2021 <https://axios-http.com/docs/intro>

Bain, L. 2016. ReactJs: Props vs. State. Published 27.11.2016. Read 27.08.2021 <https://lucybain.com/blog/2016/react-state-vs-props/>

Bootstrap Cheat-Sheet, n.d. Bootstrap 4 cheat sheet. Read 06.09.2021. <https://hackerthemes.com/bootstrap-cheatsheet/>

Bootstrap Library. n.d. Get started with Bootstrap, the world's most popular framework for building responsive, mobile-first sites, with jsDeliver and a template starter page. Read 27.08.2021. <https://getbootstrap.com/docs/5.1/getting-started/introduction/#community>

Chore, C. n.d. Node.js proxying made simple. Read 26.08.2021 <https://github.com/chimurai/http-proxy-middleware>

Context API n.d. Context provides a way to pass data through the component tree without having to pass props down manually at every level. Read 06.09.2021. <https://reactjs.org/docs/context.html>

Dodds, k. 2020. Application State Management with React. Published 21.07.2020. Read 06.09.2021. <https://kentcdodds.com/blog/application-state-management-with-react>

Eze, P. 2019. Building reusable UI components with React Hooks. Published 03.12.2019. Read 27.08.2021 <https://blog.logrocket.com/building-reusable-ui-components-with-react-hooks/>

File Structure System. N.d. File Structure Read 27.08.2021 <https://reactjs.org/docs/faq-structure.html>

Flexbox and Grid System. n.d. Layout: FlexBox grid system. Read 06.09.2021 <https://boosted.orange.com/v4-alpha5/layout/flexbox-grid/>

Font Choice. 2021. Fonts and Typography. Published 06.05.2021. Read 27.08.2021. <https://helpx.adobe.com/illustrator/using/fonts.html>

HTTP-proxy-middleware Library. n.d. Node.js proxying made simple. Configure proxy middleware with ease for connect, express, browser-sync, and any more. Read 27.08.2021. <https://github.com/chimurai/http-proxy-middleware>

Huergo, F. 2019. Stateful and Stateless components in React. Published 18.04.2019. Read 27.08.2021. <https://programmingwithmosh.com/javascript/stateful-stateless-components-react/>

Kunwar, S. 2018. Loading third party libraries in ReactJS. Published 05.12.2018. Read 27.08.2021. <https://sumn2u.medium.com/loading-third-party-library-in-reactjs-b01c049df8d5>

Leite, L. 2021a. Networking and empowering education stakeholders from Southern Africa. Published 25.05.2021. Read 27.08.2021 <https://eduix.com/blog/2021/05/28/networking-and-empowering-education-stakeholders-from-southern-africa/>

Leite, L. 2021b. Edupreneurs: Networking and Empowering Education Entrepreneurs Towards a Resilient EdTech Ecosystem in Southern Africa. Published 18.08.2021. Read 25.08.2021. <https://eduix.com/blog/2021/08/18/welcome-to-edupreneurs-final-event/>

Leite, L. 2021c. Lunch event of Edupreneurs. Published 31.05.2021. Read 27.08.2021. <https://blog.glowdom.com/2021/03/17/launch-event-of-edupreneurs/>

Lighthouse extension n.d. Lighthouse is an open-source, automated tool for improving the quality of web-pages. Read 06.09.2021. <https://developers.google.com/web/tools/lighthouse>

Liu, J. n.d. SWR – React Hook for data fetching. Read 27.08.2021. <https://swr.vercel.app/docs/getting-started>

Munro, L. 2019. The Role of Color in Product Design: UX of Color Palettes. Published 11.08.2019. Read 27.08.2021. <https://helpx.adobe.com/illustrator/using/fonts.html>

Nathan, K. n.d. The brand new Bootstrap Icons library to use as React components. Read 27.08.2021. <https://github.com/ismamz/react-bootstrap-icons#readme>

Rascia, T. 2018. An Overview and Walkthrough. Published 20.08.2018. Read 25.08.2021. <https://www.taniarascia.com/getting-started-with-react/>

React Bootstrap Library. n.d. Learn how to include React Bootstrap in your project. Read 27.08.2021. <https://react-bootstrap.github.io/getting-started/introduction/>

React Library n.d.a. This page is an overview of the React documentation and related resources. Read 25.08.2021. <https://reactjs.org/docs/getting-started.html>

React Library n.d.b. This page contains a detailed API reference or the React component class definition. Read 27.08.2021. <https://reactjs.org/docs/react-component.html>

React Library n.d.c. The ReactDOMServer object enables you to render components to static markup. Read 27.08.2021. <https://reactjs.org/docs/react-dom-server.html>

React Library n.d.d. Adding lifecycle methods to a class. Read 27.08.2021. <https://reactjs.org/docs/state-and-lifecycle.html#adding-lifecycle-methods-to-a-class>

React Library n.d.e. Handling events with React elements is very similar to handling events on DOM elements. Read 27.08.2021  
<https://reactjs.org/docs/handling-events.html#gatsby-focus-wrapper>

React Library n.d.f. Inline If with logical '&&' operator. Read 27.08.2021  
<https://reactjs.org/docs/conditional-rendering.html#inline-if-with-logical-operator>

React Library n.d.g. Only call hooks from React functions. Read 27.08.2021.  
<https://reactjs.org/docs/hooks-rules.html#only-call-hooks-from-react-functions>

React Router DOM Library. n.d. React Training/React Router. Read 27.08.2021. <https://reactrouter.com/web/guides/quick-start>

Sebulon. n.d. Project Edupreneurs. Read 27.08.2021.  
<https://blog.glowdom.com/2021/05/12/project-edupreneurs/>

Vijayvargiya, S. n.d. Using Bootstrap with React Application. Read 09.09.2021  
<https://medium.com/geekculture/using-bootstrap-with-react-application-66037e808db5>