Benjamin Rogers

N10212868

5/4/2021

Graphical user interface, application, website

Description automatically generated

**2021**

**CAB230 Assignment 1 Client Side**

Contents

[Introduction: 3](#_Toc71540780)

[Purpose and description 3](#_Toc71540781)

[Completeness and limitations 4](#_Toc71540782)

[Use of Endpoints: 4](#_Toc71540783)

[/rankings 4](#_Toc71540784)

[/countries 5](#_Toc71540785)

[/factors/year 5](#_Toc71540786)

[/user/register 6](#_Toc71540787)

[/user/login 6](#_Toc71540788)

[Modules Used: 7](#_Toc71540789)

[Reactstrap 7](#_Toc71540790)

[Bootstrap 7](#_Toc71540791)

[ag-grid-react 7](#_Toc71540792)

[Chart.js 7](#_Toc71540793)

[react-chartjs-2 7](#_Toc71540794)

[React Select 7](#_Toc71540795)

[Application Design: 7](#_Toc71540796)

[Navigation and Layout 7](#_Toc71540797)

[Register Page – Navigation 8](#_Toc71540798)

[Login Page – Navigation 8](#_Toc71540799)

[Rankings & Factors and Country Details – Navigation 9](#_Toc71540800)

[Design Mock-ups 10](#_Toc71540801)

[Usability and Quality of Design 10](#_Toc71540802)

[Layout 10](#_Toc71540803)

[Font and Colours 10](#_Toc71540804)

[Usability and Limitations of Design 11](#_Toc71540805)

[Accessibility 11](#_Toc71540806)

[Technical Description: 12](#_Toc71540807)

[Architecture 12](#_Toc71540808)

[Structure: 12](#_Toc71540809)

[How it works 13](#_Toc71540810)

[Test Plan 15](#_Toc71540811)

[Difficulties and Errors 17](#_Toc71540812)

[Limit Dropdown: 17](#_Toc71540813)

[Login State: 18](#_Toc71540814)

[Charts on Factors Page: 18](#_Toc71540815)

[Table Results Message 18](#_Toc71540816)

[Future Extensions: 18](#_Toc71540817)

[User Guide: 19](#_Toc71540818)

[References: 23](#_Toc71540819)

[Appendices: 24](#_Toc71540820)

[Appendix A – Testing screen shots 24](#_Toc71540821)

# Introduction:

## Purpose and description

The purpose of this application is the allow users to view and analyse happiness survey data of different countries. The dataset is based on publicly available data released as part of the World Happiness Report and includes ranks for each country, their assigned happiness score, and a range of related factors. These factors give scores on a country’s economy, family, healthy, freedom, generosity, and trust.

#### Extra features:

A picture containing chart

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceThe application that I made includes a module called react-select that I implemented to provide better usability for the dropdown menus. Instead of just having a traditional dropdown menu, react-select allows the user to search with text and autocompletes words. The dropdown will display ‘No Options’ if the option doesn’t exist. React-select also provides the user with the ability to clear the search back to the default option by clicking the cross button. It was for these reasons that react-select was used.

On the rankings page, if a user clicks on a country in the table, they will be directed to a country details page where they can see all the data for that country. This page also shows a chart comparing the happiness score over the years. This was only added to the rankings page because there were multiple results for each country.

Graphical user interface, application, website, Teams

Description automatically generatedIf a user that is not logged in selects the factors page, they are redirected to the login page with a popup that says, ‘You must be logged in to view this page!’. This was a nice feature to have instead of just blocking the page or redirecting to the login form without any indication of what happened. The navbar also has conditional rendering to show the logout button and hide the login and register buttons.

Lastly, on the factors page instead of displaying many charts on one page, users can select the factor they wish to see from a dropdown and the chart will update accordingly. This made the factors page a lot cleaner and more responsive on mobile screens. All pages are also responsive.

## Completeness and limitations

All the required functionality was implemented. Tables, charts, all endpoints and their query parameters, login/register functionality, react-router, react forms, and styling with bootstrap was implemented. Server-side filtering was used for most searches, but client-side filtering was used for the charts on the factors page so that the user could select the chart they wanted to view without making changing the data of the table. Error handling was implemented for server errors, page not found errors, and form errors.

# Use of Endpoints:

## /rankings

Graphical user interface, application, website, Teams

Description automatically generated

The ‘/rankings’ endpoint is used in the rankings page on the website and displays a list of countries, along with their rank, score, and year. The endpoint takes the year and country as query parameters. By default, the page loads with countries set to all and the year set to 2020.

## /countries

Graphical user interface, application

Description automatically generated

The ‘/countries’ endpoint is used on both the rankings and factors pages to populate one of the dropdown menus with the country names. Selecting these country names in the dropdown will update the table data.

## /factors/year

Graphical user interface, application, website, Teams

Description automatically generated

The ‘/factors/year’ endpoint is used on the factors page to display the happiness factors data for each country, and for each year. This endpoint has three query parameters which are year, limit, and country. The year defaults to 2020 whereas country and limit are both defaulted to show all. Users can filter using the react-select dropdown menus.

## /user/register

Graphical user interface, application

Description automatically generated

The ‘/user/register’ endpoint is used in the registration page to create a new user. The endpoint takes no query parameters, but the request uses a body that contains the email and password that the user is registering with.

## /user/login

Graphical user interface, application, website

Description automatically generated

The ‘/user/login’ endpoint is used on the login page so that a user can sign in to the site and access the factors page. The endpoint takes no query parameters, but the request uses a body that contains the user’s email and password.

# Modules Used:

## Reactstrap

Stateless React components for Bootstrap 4.

<https://reactstrap.github.io/>

## Bootstrap

Module to provide styling and responsive layouts. Installed alongside Reactstrap as Reactstrap does not include Bootstrap CSS.

<https://getbootstrap.com/>

## ag-grid-react

Module to provide fully featured tabled components, including sorting and filtering.

<https://www.ag-grid.com/react-grid/>

## Chart.js

Module to provide flexible charting.

<https://www.chartjs.org/>

## react-chartjs-2

React wrapper for Chart.js

<https://www.npmjs.com/package/react-chartjs-2>

## React Select

Flexible select input for ReactJS with multiselect, autocomplete, async and creatable support. Built-in ability to clear the search.

<https://react-select.com/home>

# Application Design:

## Navigation and Layout

The site has a clear, consistent navigation bar on the top which holds links for the rankings page, factors page, login page, and register page. When the user logs in, the navigation bar is updated to reflect this and replaces the login and register buttons with a logout button. The user is only able to navigate to the factors page after logging in. Clicking on the factors page if they have not logged in just redirects them to the login page.

### Register Page – Navigation

Graphical user interface, application

Description automatically generated

When the user registers, they are prompted with a success message and a link that directs them to the login page.

### Login Page – Navigation

Graphical user interface, application, website

Description automatically generated

Upon logging in, the user is automatically redirected to the factors page.

### Rankings & Factors and Country Details – Navigation

Graphical user interface, application, website, Teams

Description automatically generated

After logging in, the user is redirected to the factors page which is now available. The navbar updates to show the logout button and hides the login and register buttons.

Chart

Description automatically generated

On the pages with tables, the user can scroll through the data and also use the pagination down the bottom. The tables and charts are also responsive on mobile screens.

A picture containing chart

Description automatically generated

Clicking a table row on the rankings page redirects the user to the country details page where they can view more information on a country. The user can press the back button to get back to the previous page.

### Design Mock-ups

Below are some of the design mock-ups that the final website was based off. The layout remains consistent in the final version, with minor tweaks to elements such as the search bar.

Graphical user interface

Description automatically generatedGraphical user interface, application

Description automatically generated

## Usability and Quality of Design

### Layout

The website is logically laid out with a navigation bar at the top, a jumbotron below that, the content of the site following that, and then the footer. The navigation bar and footer are consistent on all pages. Users can expect navigation of the site to work like most sites. The navigation bar also provides the user with feedback by underlining the link they clicked. Each link is distinct from other links and has a clear and intuitive label. The register and login buttons are on the far right-hand side of the screen. The logo is also a link that navigates the user back to the main rankings page. The jumbotron is used on the rankings and factors page to distinguish between the two, each having its own SVG graphic. Components on the site are well-positioned and utilise Bootstrap’s grid system so that they have adequate space between each other. The footer is full width like the navigation bar and is still aligned with the content in the center of the page. The footer also has supplemental navigation. Links like the ‘rankings’ and ‘factors’ direct to their designated pages but there are also some dummy links for future pages.

### Font and Colours

The three main colours used are a blue, an orange/red, and a yellow which complement each other nicely. The font used for the site is Montserrat, and only one font was used to keep the pages looking consistent. Montserrat is a sans-serif font that was chosen to provide a simple, clean look to the site. For paragraph text, a font size of 16px was used across the site so that text can still be read clearly on mobile devices. Headings were used sparingly. H1’s was only used for the main headings in the jumbotron.

### Usability and Limitations of Design

Overall, the design of the website is very user-friendly and easy to use. One compromise of the design is that the charts on smaller screens get resized to be quite small. This could be improved by changing the height of the bars to be larger on smaller breakpoints, so the chart data is clearer.

## Accessibility

|  |  |  |
| --- | --- | --- |
| **Priority 1 Accessibility Requirements** | | |
| **Requirement** | **Done (yes/no)** | **How it would be implemented?** |
| Provide a text equivalent for every non-text element – alternatives to images, symbols, scripts, graphical buttons, sounds, audio and video files, and so on. | Yes (for images)  No (for charts) | The images on the site all have alt tags with appropriate descriptions. Chart.js charts are kept within a canvas, so it is up to the developer to make them accessible. Accessibility can be added with the ARIA attributes on the canvas element or added using internal fallback content placed within the opening and closing canvas tags (Chart.js.org, 2021) |
| Ensure that all information conveyed with color is also available without color, for example from context or markup. | Yes | Information that is given colour will still be accessible if the user cannot see colour (W3C, 2000). The coloured buttons on the site are still bigger and emphasized more than the regular links. Overall, the website uses a good contrast ratio so that if a user is colour blind, they can still distinguish a coloured button on a coloured background. Footer links are white and have effective contrast against the orange background. The main navigation does not rely on colour and has all elements on a white background so that is easily usable. Information like the ‘Register here’ link on the login page could be more distinguishable since the only way someone can tell it is a link, is because it is blue. Instead, this could be given an underline and perhaps an icon to clearly show that it is clickable. |
| Organize documents so they may be read without style sheets. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document. | No | To do this, content would have to be organised within more divs in the HTML so that if no vertical positioning styles are applied in the CSS, it would still structure the content to be read line by line if CSS was turned off (WC3, 2000). |
| Ensure that text equivalents are updated when dynamic content changes. | No | If dynamic content such as images change, then the alt tag for the images should change appropriately (W3C, 2000) |
| Avoid causing the screen to flicker. | Yes | Content that flickers will be avoided entirely for the website (W3C, 2016). W3C provides a precise formula for calculating safe general flash thresholds but flickering content will not be added to this site. |
| Use the clearest and simplest language appropriate for a site's content. | Yes | The language used across the site is simple and easy to understand. This is particularly important for the navigation. The site has clear and accurate headings and menus. Paragraph text is limited to one main idea. The use of passive voice is avoided. Specialized or unfamiliar vocabulary is avoided. Complex sentence structure is avoided. |
| For tables, identify row and column headers – clearly differentiated from the data. | No | Whilst the tables in the site have column names along the top. Since the table does have a lot of data, if the user scrolls down it can become unclear what data corresponds to what column name. The table height should be limited so that the column names are in view when browsing the data (W3C, 2016). |

# Technical Description:

## Architecture

### Structure:

Text

Description automatically generated with low confidenceThe organisation of the src directory for the application can be seen on the right. There are 4 main folders – assets, components, helper, and pages. The assets folder is for storing images that are used when the app is compiled. This is for the main SVGs and logo files. The favicon for the site is instead found in the public folder. The main App.js and index.js files are kept in this main src directory as per create-react-app’s default setup. The api.js file is also kept in here so that is accessible across the entire app.

Text, letter

Description automatically generatedThe components folder is for storing react components that can be imported across the program. Each component was organised in its own folder as some have CSS files, so it was cleaner this way. They could have just been all kept in the one folder but as the project grows and more custom styling is added, having a separate folder for each component would cleanly seperate CSS files.

Graphical user interface, application

Description automatically generated with medium confidenceThe helper folder contains one file called Context.js that is used for maintaining a login context across the application. It was put inside a helper folder so that if any other helper functions need to be written in the future, they can be kept in a central location.

The pages folder contains components that map to a URL. These are the components that react-router renders rending on the route. Some error pages for 403 and 503 errors are also kept here. A custom protected route was also created and stored here.

### How it works

Within App.js, this is where the routes for the application are specified and the login context is shared between the components. React router is used within this page to render components according to their route. The protected route is used in App.js so that the user cannot access the factors page without being logged in.

Text

Description automatically generated In api.js, this is where the main API calls are stored to get row data for both the rankings page, and the factors page. There are two functions ‘getCountryRankings’ and ‘getCountryFactors’ which return a fetch with the data. Then there is a function called ‘useCountryData’ which takes in a route parameter for either using the factors or rankings data. This function returns the data, a Boolean if the request is loading, and also any error message that might be received from the server.

Text

Description automatically generatedText

Description automatically generatedThe rest of the logic can be found in the CountryRankings.js and CountryFactors.js components. These render a search bar, a table, a chart and a modal. They also have some conditional rendering for elements such as the results message.

The SearchBar.js component is quite involved and houses state to manage several things that can be seen to the right. The search bar component uses the ‘/countries’ endpoint to populate the dropdown with country names and uses a lot of state for the react-select menus. There are 3 functions for toggling clearing each input and useEffect hooks for updating the limit options dropdown so that it gets updated every time a new year is selected. The search bar changes the data of the table and charts by setting the state of the search country, search year and search limit in the CountryFactors.js file and CountryRankings.js file. The props that are passed to the search bar can be seen above. These are then the values that are passed to the ‘useCountryData’ function that interacts with the API.

Text

Description automatically generated

The rowData that is returned from ‘useCountryData’ is then used for the tables and charts. The chart uses client-side filtering to show just the top 15 results.

## Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Expected Outcome** | **Result** | **Screenshot/s**  **(Appendix A)** |
| Search all countries and all years | Results displayed, chart hidden, and result message changed. | PASS | 01 |
| Search all countries and individual years | Results displayed, chart shown, and result message changed. | PASS | 02 |
| Search countries with junk text | No result displayed in table. Dropdown returns ‘No Options’. | PASS | 03 |
| Search year with junk text | No result displayed in table. Dropdown returns ‘No Options’. | PASS | 04 |
| Search limit with junk text | No result displayed in table. Dropdown returns ‘No Options’. | PASS | 05 |
| Search all countries with an individual year and all limit selected **(factors page)** | Results displayed and chart visible (if results >= 15). Results table should show maximum number of rows for that year. | PASS | 06 |
| Search all countries with an individual year and limit set to 1 **(factors page)** | Results displayed, result message updated, and chart hidden. Table should also scale down to smaller size. | PASS | 07 |
| Search all countries with an individual year and limit set to maximum **(factors page)** | Results displayed, result messaged updated and chart visible. Selecting the maximum limit option will give same result as selecting ‘All’. | PASS | 08 |
| Search individual country with individual year **(factors page)** | Results display 1 row, result message updated, and chart hidden. Limit options input should also be disabled. | PASS | 09 |
| Search country with a year that does not have any data. E.g. Angola in year 2020. **(factors page)** | Results display no data, result message updated to show 0 and error modal popup on screen with message. | PASS | 10 |
| Clear search country with cross button **(factors page)** | Results should show all countries with the values that were set previously for year and limit | PASS | 11 |
| Clear search country with cross button **(factors page)** | Results display data with year defaulted back to 2020, countries set to previous value and limit set to all. | PASS | 12 |
| Clear search limit with cross button **(factors page)** | Results display data with limit set to all, and countries and year set to the previously used values. | PASS | 13 |
| Clear search country when only one row is shown **(factors page)** | Results display data with all countries and limit input is now interactable again. | PASS | 14 |
| Select a limit, change the year, and then view the limit options again **(factors page)** | Limit options update according to the year set. | PASS | 15 |
| Search individual country with all years **(rankings page)** | Results display data for all years from 2015 – 2020 for that country or the years available. Update results message and hide chart. | PASS | 16 |
| Register with fields empty | Display error in Bootstrap alert | PASS | 17 |
| Register with one field empty | Display error in Bootstrap alert | PASS | 18 |
| Register with email and password entered | Display successful registration message in Bootstrap alert. Message should include ‘Click here to login’. | PASS | 19 |
| Click ‘Click here to login’ after successful registration | Redirects to login page | PAGE | 20 |
| Login with field empty | Display error in Bootstrap alert | PASS | 21 |
| Login with one field empty | Display error in Bootstrap alert | PASS | 22 |
| Login with incorrect email and/or password | Display error in Bootstrap alert indicating that either the email or password is incorrect | PASS | 23 |
| Login with correct email and password | Redirect to factors page and update navbar to show logout button. Hide the login and register buttons from navbar. | PASS | 24 |
| Click logout button | Toggle modal with message and logout button. If this second button is clicked, then user is redirected to login page and navbar is updated. Factors page is then inaccessible until they login again. | PASS | 25 |
| Click rankings link in navbar | Redirect to rankings page | PASS | 26 |
| Click factors link in navbar | Redirect to factors page if logged in otherwise redirect to login page | PASS | 27 |
| Click login link in navbar | Redirect to login page | PASS | 28 |
| Click register button in navbar | Redirect to register page | PASS | 29 |
| Click logo in navbar | Redirect to rankings page | PASS | 30 |
| Click ‘Register here’ link on login page | Redirect to register page | PASS | 31 |
| Select factors from dropdown at the bottom of factors page | Show chart corresponding to factor | PASS | 32 |
| Click rank column heading in tables | Sort table data in opposite order according to rank | PASS | 33 |
| Click links that do not have pages yet | Refresh page but site does not crash | PASS | n/a |
| Click table row in rankings page | Redirect to country details page with table and chart data for all years available for that country | PASS | 35 |
| Click back button on country details page | Redirect back to rankings page | PASS | 36 |
| Country details page is still loading | Display heading that says ‘Loading…’ until data is fetched | PASS | 37 |
| Go to page that does not exist | Display custom page for 404 error with button that goes back to rankings page. | PASS | 38 |
| Server error | Display custom page for 503 error with button that goes back to rankings page. | PASS | 39 |

## Difficulties and Errors

### Limit Dropdown:

One of the challenges I had with the limit dropdown was that the values should update when a different year is selected and not update when the row data changes. I had an issue where I was relying on the one API call for the table data and the limit dropdown, but the options would disappear or change every time the table changed. To fix this I only use the row data for the initial range of values in the limits array. However, they are updated with a separate fetch statement. This can be seen in the search bar component file.

### Login State:

Another challenge I faced when building the application was trying to implement conditional rendering of the navigation bar when the user is logged in. Initially, all I was doing was using a ternary statement in JSX to check for a token, but this would not render the change straight away and would only work if the page was refreshed. To fix this, I created a helper file which houses a login context that was used to App.js file and the navigation bar component. This passes the login state to all other components so that they can check and set the current login state whenever needed. I had to use the useContext hook for this to work.

### Charts on Factors Page:

Initially, I had the factors page just show all 6 charts for the individual factors one underneath another. However, I thought this was a clumsy design so I used client-side filtering to store all the factors data for each factor in its own array that could be toggled with an input field. This made the charts functionality a lot cleaner.

### Table Results Message

Depending on the results of the search, the results message is different so that it makes more sense grammatically. Initially, I had a bunch of long ternary operators stringed together in the JSX to deliver this functionality, but it was very messy. Instead, I opted to have the conditions in the function body that set a state for the result message. This worked out a lot better and was much more readable.

# Future Extensions:

Future improvements to the site could include adding corresponding country flag icons for the data. Since this is not available in the current API, another API would likely need to be used. More detailed charting could be added where the user could toggle types of charts such as bar or line charts. The site could also organise that data into regions and compare specific regions in the charts. This would be interesting for analysing data specific to Asia or Europe for example. The tables could also be updates to colour certain rows based on conditions. If a country ranks first, then the row could be coloured to show it. Also, if a happiness score is below 4 for example, then the row could be coloured to show it. More sophisticated forms on the register page could also be used so that the form only submits if the password meets the strength requirement.

# User Guide:

The rankings page is the starting page of the application. Here the user can view country rankings without needing to sign in. A chart with the currently selected year can also be found below the table. The search inputs can be used to change the country or year on this page. The inputs also accept text and will autocomplete. Searches can also be cleared using the cross that appears. To proceed, select the register button in the top right to create a new account.

Graphical user interface, application, website

Description automatically generated

Chart, bar chart

Description automatically generated

Once on the register page, the user can enter an email and password that they wish to make an account with. The form with give a success message or an error message so the user understands what happened. Once they have registered, they can click the ‘Click here’ link to go to the login page.

Graphical user interface, application

Description automatically generated

Once on the login page, the user can log in using their email and password that they registered with. The form will redirect the user to the factors page or display an error if the credentials are incorrect.

Graphical user interface, website

Description automatically generated

If a user clicks the factors page link before logging in, then they are redirected to the login page and prompted with a message.

Graphical user interface, website

Description automatically generated

Once logged in, the user can browse the factors page that was previously blocked. The navigation bar also updates to show a logout button. On this page, there is also a third input on the search bar called limit. This can be used to limit the number of search results. The tables on both pages can have pagination controls in the bottom right corner for larger sets with over 40 results. Clicking the ‘Rank’ column of the table will also sort rows in descending order according to rank.

Graphical user interface, application, website

Description automatically generated

Below the table on the factors page, there is a chart that changes depending on what factor is chosen. This chart can show each happiness factor for the top 15 countries, depending on the year that was selected in the main search bar. Users can also hover over a bar in the chart to view the value.

Chart, bar chart

Description automatically generated

On the rankings page, if a user clicks on a country row in the table it will redirect them to a country details page.

Graphical user interface, application, website

Description automatically generated

On the country details page, the user can see country data for all available years and a chart to show it as well. There is also a back button that will direct them back to the rankings page.

Graphical user interface

Description automatically generated

# References:

Chartjs.org. (2021). Accessibility | Chart.js. Retrieved 10 May 2021, from <https://www.chartjs.org/docs/latest/general/accessibility.html>

WC3. (2016). Understanding WCAG 2.0. Retrieved 10 May 2021, from <https://www.w3.org/TR/UNDERSTANDING-WCAG20/Overview.html#contents>

W3C. (2000). Examples: WAI Web Content Accessibility Curriculum - slide "2.1 - Ensure that all information conveyed with color is also available without color, for example from context or markup.." Retrieved 10 May 2021, from <https://www.w3.org/WAI/wcag-curric/sam25-0.html>

W3C. (2000). Examples: WAI Web Content Accessibility Curriculum - slide "6.1 - Organize documents so they may be read without style sheets. When an HTML document is rendered without associated style sheets, it must still be possible to read the document.." Retrieved 10 May 2021, from <https://www.w3.org/WAI/wcag-curric/sam52-0.htm>

W3C. (2000). Examples: WAI Web Content Accessibility Curriculum - slide "6.2 - Ensure that equivalents for dynamic content are updated when the dynamic content changes.." Retrieved 10 May 2021, from <https://www.w3.org/WAI/wcag-curric/sam54-0.htm>

# Appendices:

## Appendix A – Testing screen shots

Graphical user interface, application

Description automatically generatedTest 01

Test 02

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generatedTest 03

Test 04

Graphical user interface, application

Description automatically generated

Test 05

Graphical user interface, application, Teams

Description automatically generated

Test 06

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, application

Description automatically generatedTest 07

Test 08

Graphical user interface, application

Description automatically generated

Test 09

Graphical user interface, website

Description automatically generated

Graphical user interface, application, website

Description automatically generatedTest 10

Test 11

Graphical user interface, application

Description automatically generated

Test 12

Graphical user interface, application

Description automatically generated

Test 13

Graphical user interface, application, Teams

Description automatically generated

Test 14

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, application, Teams

Description automatically generatedTest 15

Test 16

Graphical user interface, application, Teams

Description automatically generated

Test 17

Graphical user interface, application

Description automatically generated

Test 18

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generatedTest 19

Test 20

Graphical user interface, website

Description automatically generated

Graphical user interface, website

Description automatically generatedTest 21

Test 22

Graphical user interface, website

Description automatically generated

Graphical user interface, website

Description automatically generatedTest 23

Test 24

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, text, application

Description automatically generatedTest 25

Test 26

Graphical user interface, application, website

Description automatically generated

Graphical user interface, application, website

Description automatically generatedTest 27

Graphical user interface

Description automatically generatedTest 28

Test 29

Graphical user interface, application

Description automatically generated

Graphical user interface, application, website

Description automatically generatedTest 30

Test 31

Graphical user interface, application

Description automatically generated

Chart, bar chart

Description automatically generatedTest 32

Test 33

Graphical user interface, table

Description automatically generated

Graphical user interface

Description automatically generated with medium confidenceTest 35

Test 36

Graphical user interface

Description automatically generated

Test 37

Graphical user interface

Description automatically generated with medium confidence

Test 38

Graphical user interface, text, website

Description automatically generated

Test 39

Graphical user interface, website

Description automatically generated