Smiling Globe - An Interactive Webapplication for Happiness Data

Rui Zhang[*] BSc Computer Science
Tamàs Nemes[†] BSc Computer Science
Tianjian Yi[‡] BSc Computer Science
Ian Wimmer [§] BSc Computer Science

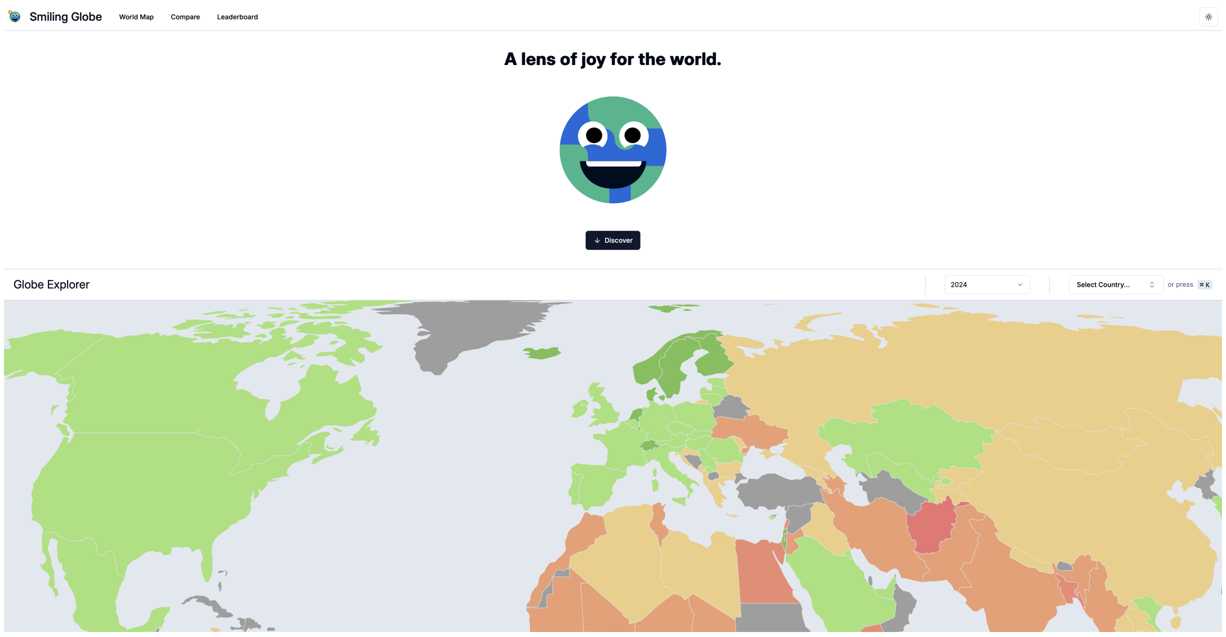


Figure 1: The Landing Page of Our Webapplication

Abstract

The Smiling Globe project presents an interactive web application designed to provide intuitive access to data from the World Happiness Report. By offering features such as a global heatmap, detailed country comparisons, and a leaderboard of happiness rankings, the platform enables users to explore and analyze happiness metrics efficiently. Built with a focus on user-friendliness and accessibility, the application leverages modern web development tools like NextJS, TailwindCSS, and Supabase to deliver a seamless experience. Future enhancements aim to include localization, backend optimizations, and AI-assisted insights to broaden functionality and engagement.

1 Problem Statement

As the world advances in its steps, the focus on the individual and its feelings becomes more and more emphasized. The development of faster and more accessible public transport opportunities lead to an increase in migration, as people increasingly wonder which places on this earth will allow them to live their life to the fullest. Of course, for many, one of the determining factors is how happy a place and the residents of that place are. As such, interest in data such as those provided by the World Happiness Report surges.

[*]e-mail: zhangrui@student.ethz.ch
[†]e-mail: tnemes@student.ethz.ch
[‡]e-mail: tianyi@student.ethz.ch
[§]e-mail: iwimmer@student.ethz.ch

This is only one of many reason demonstrating the existence of widespread interest in the data of the World Happiness Report. For many individuals, navigating and reading through pages upon pages of content to find the specific data they are looking for will seem tedious. Our project, the Happiness-Explorer, is meant to solve this problem by providing an interactive and intuitive summary of the most important data from the World Happiness Report such that interested individuals may find what they need efficiently.

2 Designing the Project

2.1 Tools

Before any coding, we created a preliminary design of the project. This allows us establish at the very beginning what we will have to implement. It separates the design of this application from the actual implementation process. Furthermore, it allows us to quickly produce a mockup that we could communicate to our teaching assistant, enabling early feedback before we create anything suboptimal. Designing was done in Figma due to the expertise of our team members in this tool and its widespread usage.

In this same stage of our project, we voted for the use of 'shadcn/ui'. It had been clear that we wanted to use a user-interface-library for this application - Not only does it simplify the creation process heavily, it also offers uniformity for our user interface. The reason we settled for shadcn is the high customizability it offers and how lightweight it is. The shacn command line interface allows for the addition of individual components to our projects repository instead of the installation of all of the components at once, which is the way many traditional UI-libraries handles it. Moreover, this CLI, instead of importing the components into the 'node_modules' folder of our project, does so into the 'components/ui' folder. This makes the individual components' code easily accessible and open to our own modification. Another reason in favor of the decision of using shadcn is its ui-kit in Figma, which allowed us to design our mockup in Figma directly with the shadcn components.

2.2 Figma Mockup

2.2.1 Index Page

Every website needs an index page, and since our target audience is interested in exploring the World Happiness Report data, we decided to make the index page a dashboard that highlights the main insights. However, we felt it would be unintuitive to leave the index page as just a dashboard without any context or explanation. So, we added a brief introductory sentence / slogan at the top to explain the purpose of the website - "A lens of joy for the world."

Below this section is the main dashboard, designed to present the data in an intuitive way for casual users. We wanted to avoid simple lists or grids of data, which we felt would be overwhelming. Instead, we opted for a heatmap of the world, providing a clear overview of global happiness and allowing users to click on countries for more details.

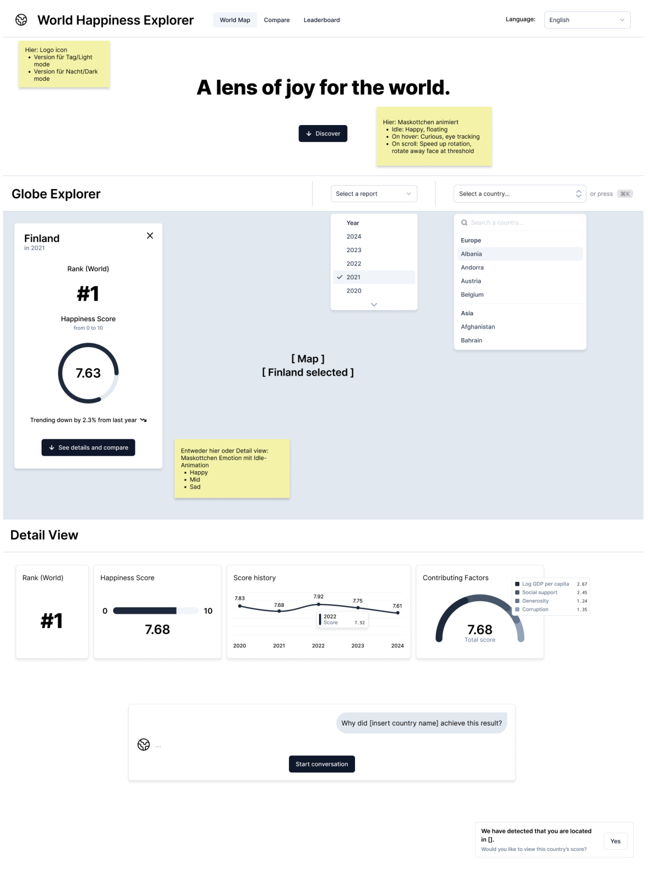
To improve accessibility, we included a search bar for users to select a country using only the keyboard. When a country is selected, its happiness data should be displayed. There are several ways to do this. One option is to show the data as an overlay on the map, visually linking the data to the selected country. However, this could cause clutter, especially on mobile. Another option is to display the data in a separate section below the map, which avoids clutter but adds an extra scroll. To balance these, we decided on a hybrid approach: a simple overlay showing just the happiness score, with more detailed information below the map. For mobile, the overlay and detailed section are positioned below the map to keep the layout clean and easy to navigate.

2.2.2 Compare Page

We considered that a dashboard designed like above would not account for the curiousness of some individuals. Very important insights are also gained from comparing two countries and their happiness data with each other. One possibility is to add another piece of functionality into the same index page allowing for multi-select of countries. Since we want to make the application as intuitive as possible, we figured that simplicity is key. Henceforth, we opted for an entire separate page for the comparison functionality. This also enables a better division of responsibilities when actually implementing the website, since we would have an entire other page. The weakness of this approach was the same as its strength. Another page means another navigation step. Our decision was based on the assumption that a navigation step is simpler than a multi-select option in the world-map.

For the comparison page itself, we decided on the ability to select two countries and display their data side-by-side. Our first solution was showing the data of the two countries below one another, essentially having two rows of data. The alternative would be two columns of data. The advantage of two rows is that the responsiveness of the website is guaranteed without much additional overhead in implementation, as we could make the data horizontally scrollable. Furthermore, we could reuse the same component in the exact same way on the index-page. The alternative was settled for after the meeting with our teaching assistant, due to its visual simplicity. We could still reuse this component, but just allow for a different flex-direction in the implementation. Adjusting for responsiveness then meant adjusting for the same flex-direction

We also wanted users to be able to compare the development of happiness between two countries. To this end, we added two charts at the top of the page. One would contain the two graphs representing the development of the happiness scores of both countries. One would do so for the ranking of the two countries.



Hier: Logo icon
- Version für Tag/Light mode
- Version für Nacht/Dark mode

Hier: Maskottchen animiert
- Idle: Happy, floating
- On hover: Curious, eye tracking
- On scroll: Speed up rotation, rotate away face at threshold

Entweder hier oder Detail view: Maskottchen Emotion mit Idle-Animation
- Happy
- Mid
- Sad

Why did [insert country name] achieve this result?

We have detected that you are located in [].
Would you like to view this country's score?

Figure 2: Initial Figma mockup of the index page

2.2.3 Leaderboards Page

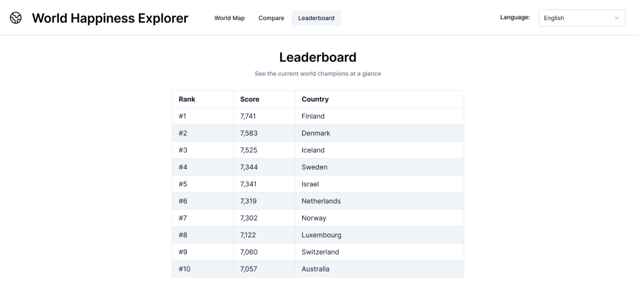
As a final feature, we designed the leaderboards page of the webapplication, for those interested in only the best. For the purpose of reusability, uniformity and implementation simplicity - especially considering the limited time we have for this project, we would show on this leaderboards page the top 10 happiest countries and their data by positioning them in rows and reusing the same component for data representation as we did for compare and index pages.

Lastly, to make the website more friendly and welcoming, we also designed a mascot for the website - a smiling globe.

3 Implementation

3.1 Tools

The development of this webapplication started with many choices, especially the technology we would want to use to implement this project. Our web application would consist of a backend and a frontend. To implement the frontend / UI of the application, which will be exposed to the user, we settled for NextJS, a framework designed for full-stack interactive webapplications, which fits our goal. Additionally, NextJS uses React under the hood. Since we had learned react in the lectures, the transition to NextJS would not be as harsh as other technologies such as Vue or Angular. Shadcn, our chosen UI-library, also offers easy integration with NextJS. However, other options still remain. For instance, we could also have used the React, ExpressJS stack we have used in our assignments. The decision for NextJS still persisted, since setup and deployment of NextJS applications are easily realized. Additionally, some of our team members were already acquainted with development with this framework.



Leaderboard

See the current world champions at a glance

Rank	Score	Country
#1	7,741	Finland
#2	7,583	Denmark
#3	7,525	Iceland
#4	7,344	Sweden
#5	7,341	Israel
#6	7,319	Netherlands
#7	7,302	Norway
#8	7,122	Luxembourg
#9	7,060	Switzerland
#10	7,057	Australia

Figure 3: Initial Figma mockup of the leaderboards page

We also opted for tailwindcss, a CSS Framework to reduce the overhead required to write functioning CSS code. It is also has seamless integration into shadcn.

Since NextJS is a fullstack framework, it also offered backend capabilities, but no immediate database. To house the data we have from the world happiness report, we chose Supabase. Supabase has the advantage that it offers an api out of the box. That way, a lot of work is simplified for us and management of the immense amount of data the reports have is done efficiently. Many other backend-as-a-service options still exist though, such as Firebase. Firebase also provides many other services such as real-time databases, and serverless features. These advantages of firebase were exactly the disadvantages in this use-case: Our project needed a static backend with all the information from the world happiness report. Therefore, features such as the real-time-database are redundant. Another reason against the use of Firebase is its increased complexity compared to Supabase, as some of our team members noticed in a previous project.

3.2 Programming

We split up our work such that each person have an equal amount of tasks. Ian implements the customizable detailed data display component which would be reused in all three pages. Thomas writes the code for the comparison page. Tamas codes the leaderboards page, the Supabase integration and creates the mascot with Rive, an animation production software and Rui produces the index page along with the navigation header.

3.2.1 Detailed Data Display

The detailed view of all of the happiness data of a country is written in the files country-detailed-view.tsx, custom-radial-bar-chart.tsx, custom-line-chart.tsx, and custom-card.tsx, all found in the 'components/ui' folder of our project. The main component is situated in country-detailed-view.tsx. This component, the 'CountryDetailedViewContainer' requires the happiness data of the countries to be passed, including name, flag emoji, happiness score, score history, etc. To account for the fact that this component will be displayed horizontally on leaderboards and index pages but vertically on the compare page, the 'adjust_on_large_device' boolean property exists. Depending on which properties were passed, the component would also show a variable amount of information. To realize this, Ian separated the component into multiple cards just as specified in our design. These cards can be found in the 'custom-card.tsx' file.

3.2.2 Layout

The layout of the website consists of a navigation menu and a theme provider to allow for light and dark-mode. The package next-themes provides a simple plug-in component, which can be wrapped around the body of the page and dynamically swaps out the Tailwind classes, determining which pre-defined color palette to use. This component was implemented in the top-level layout.tsx file. The navigation menu is a separate component found in components/ui/header.tsx. In its standard layout, it is a simple flexbox with the navgiation options and a logo. In a mobile setting, it changes to a clickable burger-style menu. This separation of design is done via extensive use of tailwindcss media queries. A simple useState hook from react controls the opening and closing logic of the mobile navigation menu.

3.2.3 Index Page

The app/page.tsx file is the one that is rendered when first opening the website. The code consists of the landing section with the mascot and the slogan and the detailed view section which uses the 'CountryDetailedViewContainer' component. The center piece of the code is the 'GlobeExplorer' component, found under components/globe-explorer, which exposes an 'onCountryChange' callback called each time a country is selected in the globe-explorer, allowing for state management and dynamic updating of the detailed view section. This makes the globe-explorer act almost like an input field.

The globe-explorer utilizes the react-simple-maps library at its core. It processes a json file containing the data of all countries and displays it. We use the json file found in `https://cdn.jsdelivr.net/npm/world-atlas@2/countries-110m.json`. On load, the page queries the countries and their data to display the list of countries for the search bar and the heat map. Each country in the map is its own separate component (see components/globe-explorer/geo-country.tsx) which contains logic to convert the happiness score into a color. Every time a country is selected, not only is the 'onCountryChange' callback called, the internal state of the component is also updated to change the minimized detailed view showing only the happiness score of that country.

Both the header and the minimized detailed view make extensive use of shadcn components such as Select, Popover and Command. Shadcn also offers many options for charts, one of which is also utilized to display a graphical representation of the happiness score.

3.2.4 Compare Page

The page.tsx file only consists of fetching the years and countries from the backend as using server components here makes it simpler. But the most part of the work is done in the MyComponents.tsx file for which the years and the countries are passed as props to from page.tsx. If you look at the compare page, the boxes for selecting the year and the two countries at the top of the page are implemented using the shadcn Select component. On a value change, depending on which of the Select boxes was changed, different states are changed that are used in later components.

One of them is the shadcn Chart component directly below the Select boxes. Those use the values stored in the states to fetch the corresponding data from the backend and display them in a linechart. Below that, for the side-by-side comparison of the two selected countries, we use the 'CountryDetailedViewContainer' component which we designed ourselves. This uses the information stored in various states to fetch the data accordingly. Also, for this page, the adjust_on_large_device setting in the component is set to true if the screen is large enough to enable a vertical display for better comparison in contrast to the horizontal layout on the Index Page.

3.2.5 Leaderboards Page

Again, here the page.tsx file is only used to do a first fetch of the initial data (default for year 2024) which is then passed to the client component LeaderBoard.tsx so it already shows the leaderboard for the year 2024. In that file, a Select component from shadcn is used to select the year. Each time a new year is selected, the 10 countries with the highest happiness scores are fetched from the backend. Those countries are then displayed together with additional data about their happiness scores with components that are also used in the 'CountryDetailedViewContainer' for consistency reasons. There is also a scrollbar below the cards of the countries which is realized by the shadcn 'ScrollArea' component.

4 System Showcase

On a sunny Friday evening our future user, John, sits in his garden, drinks a can of beer, and enjoys the sunset. Feeling happy and grateful, he wonders if their country is happy in general or if there are countries that are even happier than his. That's where the user hops on his laptop to research about this and finds our website 'Smiling Globe'. Entering the website, John is forwarded to our landing page, where he is greeted by our mascot, the smiling globe. There, he can select a year and a country of his choice on our World map to inspect different kinds of information of said country in the selected year. John will be able to see how his country ranks in terms of the Happiness Score and also how the Happiness Score changed over the years. Furthermore, he can also see a visual of what factors contribute to the Happiness Score e.g. Generosity or Social Support.

Now, John is interested in directly comparing his country with other countries to see the differences between those countries. This he can do by clicking on 'Compare' on the Menu with which he is forwarded to the Compare Page. There, he can again select a year and two countries, this time to have a comparison between the countries. When selected, there will be two graphs at the top of the page showcasing the development of the world rank of the two countries over the years. Moreover, there is a second graph displaying the Happiness score next to the world rank graph so that the user can visually see the changes of Happiness in those countries. Scrolling down, the user can see a side-by-side comparison of the two countries in terms of Happiness Rank, Happiness score, Contributing factors etc. With that, our user can get a bunch of information on these two countries and really see what differs. They can also switch up the countries they are comparing to satisfy their needs to compare different countries.

John has done enough comparing now. He is quite proud that his country does very well in terms of Happiness and also that they are in the top 10 happiest countries in the world. Now, he is curious about what other countries are also in the top 10 in terms of Happiness score. For that, he clicks on 'Leaderboard' on the Menu bar and is sent to our Leaderboard page. There, he can again select a year that he wants to observe and the top 10 happiest countries in that year are displayed with lots of other information about the Happiness Scores. This way, the user can see which ten countries are the happiest in terms of Happiness Score and also what constitutes to their Score all on one page. After visiting all the pages, John is quite satisfied with the insights he gained and plays around with the different functions on the various pages for a while before leaving the website, knowing that he became just a bit smarter today.

5 Discussion

5.1 WCAG Report

As part of assessing the accessibility of our website, we conducted a comprehensive evaluation against the Web Content Accessibility Guidelines (WCAG) 2.1 standards, aiming for conformance level AA. The assessment showed that our site successfully met 29 out of 50 success criteria, demonstrating significant progress toward accessibility. However, we fell short (failed) on 6 criteria, highlighting areas for improvement. Additionally, 14 criteria were deemed not applicable to our website, and 1 criterion could not be conclusively assessed due to insufficient data. (The full WCAG 2.1 report can be found in the report_poster directory on GitLab) or under https://happiness-explorer.vercel.app/smiling-globe-wcag-report-AA.html.

5.2 Limitations and Future Improvements

One limitation of our deployed product is its speed on the safari browser. Compared to other browsers, the webapplication sometimes stutters in safari. Given more time for this project, the optimization of the webapplications logic for safari browses would thus be a high priority task.

Additionally, the speed of data loading on this website, take for example the leaderboards page, is acceptable, however not optimal. As attention span and audience retention rate decreases with the rise of short form social media content, it is also imperative that we look into backend optimization with Supabase to allow for snappy data fetching.

Lookiung at the data provided by the world-happiness-report, one may notice that it offers far more data, such as the demographic composition of pieces of data. Due to time limitations, we were not able to include this data in our database. The detailed data view component already has all of the capabilities to show the demographic composition data, so this can be considered a small but attention worthy improvement in the future, since the full implementation would only entail the integration of the data into Supabase, the fetching process and the passing of all necessary arguments to the component.

Another feature left out due to lack of time is localization. Since this application is meant to be for interested individuals, the target group may be heavily international. We also have a language selector in the header of the website. Our team has already began implementation on this feature, but we were not able to finish it. This is also a possible direction we may continue our work.

Finally, one feature that had been discussed with our teaching assistant was also an AI assistant offering more insight into the data provided. An actual implementation would most likely consist of a GPT api wrapper with all of the data from the reports supplied to the AI in a prompt beforehand - the reason for this being that none of our team members have yet had the opportunity to train our own AIs. We were recommended to keep this feature as a 'good-to-have' only to be implemented if we really have time.

5.3 Lessons Learned

This project was a very big learning experience for many of us. The main lessons to be taken away are listed below:

- For some of our team members, this is the first time working with many of the technologies listed in this report. From a simple hard-skills perspective, we have therefore improved in webdevelopment significantly.

- For others, it was also one of the first times working with a group on a project, improving our soft-skills and cooperation capabilities. For instance, during the entire process, we dealt with organization questions such as which platform we would use for meetings, in the end settling for Discord.

- Throughout the meetings with our teaching assistant, we also acquired more skills in accessible and intuitive webdesign, as we were guided through our design decisions via critical questions and thinking.

- Webengineering is not simple. It requires a lot of attention to detail and cooperative effort. For a website to look fully fleshed out and polished, a month is not enough.