Development of MESOC Toolkit Web Application in React

Jermaniš, Erik

Undergraduate thesis / Završni rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:798474

Rights / Prava: In copyright/Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-10-15**

Sveučilište u Rijeci
**Fakultet informatike
i digitalnih tehnologija**

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Informatics and Digital Technologies - INFORI Repository

University of Rijeka – Department of Informatics

Erik Jermaniš

Development of MESOC Toolkit Web Application in React

Bachelor's thesis

Mentor: prof. dr. sc. Sanda Martinčić – Ipšić dipl. Ing.

Rijeka, 05.7.2021.

Rijeka, 8.5.2021.

Zadatak za završni rad

Pristupnik: Erik Jermaniš

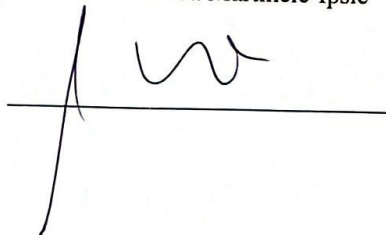Naziv završnog rada: **Razvoj MESOC web aplikacije u React-u**

Naziv završnog rada na eng. jeziku: **Development of MESOC Toolkit Web Application in React**

Sadržaj zadatka:

Cilj završnog rada je izrada web aplikacije s alatima za prikupljanje, organizaciju i prikaz analize podataka u MESOC domeni. Za analitički prikaz podataka potrebno je izgraditi interaktivno sučelje napisanog u React-u, koje će biti responzivno. Drugi dio zadatka je povezivanje sučelja s pozadinskim djelom aplikacije (backend) koji prikazuje, vizualizira i analizira podatke MESOC domene. Pisani dio završnog rada uključivati će opis procesa izrade aplikacije te opis dodatnih paketa korištenih kao pomoć u izradi same aplikacije.

Mentor

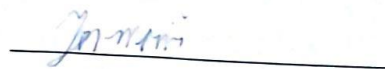Prof. dr. sc. Sanda Martinčić-Ipšić

Voditelj za završne radove

doc. dr. sc. Miran Pobar

Zadatak preuzet: datum

(potpis pristupnika)

Abstract

Application described in this thesis is a web application which serves as georeferenced visualization tool for assessing the value and impacts of cultural policies and practices. Application is written in React – a JavaScript library for building user interfaces.

The development process consists of two parts. First part consists of planning and creating the user experience and user interface, and after that, coding the application. Second task is connecting the application to a backend server that processes the data collected in the application. The backend server consists of complex NLP tasks performed in a pipeline, and serving the data to the frontend application.

Keywords: React, web application, frontend

HR Sažetak

Razvoj MESOC web aplikacije u React-u

Aplikacija opisana u ovom završnom radu je web aplikacija koja služi kao georeferencirani alat za vizualizaciju procjene vrijednosti i utjecaja kulturnih politika i praksi. Aplikacija je napravljena u React-u – JavaScript biblioteci za izgradnju korisničkih sučelja.

Proces razvoja aplikacije sastoji se od dva dijela. Prvi dio uključuje planiranje i kreiranje korisničkog iskustva te korisničkog sučelja, a nakon toga kodiranje same aplikacije. Drugi zadatak je povezivanje aplikacije sa pozadinskim poslužiteljem koji obrađuje podatke prikupljene u aplikaciji. Pozadinski poslužitelj izvodi slijed postupaka računalne analize and ulaznim tekstovima te priprema podatke za prikaz na web aplikaciji.

Ključne riječi: React, web aplikacija, frontend

Table of Contents

1. Introduction

Web application development is the process involved with building a web application. Most cases of web application development will involve defining the problem, mocking-up the solution, choosing a development tool, and finally, building the web application.

First and the most important part of development process of every application is defining the problem and creating the plan for the application development. It is not uncommon for misunderstandings to arise between the developer and the client. This happens because people tend to interpret the world through their own lenses or sense of reality. By creating a good plan, the developers are ensuring that they get on the same page with the client and that they avoid any potential misunderstandings.

There are several ways of defining a development plan, some of which are [1]:

- Creating a wireframe,
- Creating a working mock-up,
- Creating a task list,
- Defining use cases.

The second method, creating a working mock-up, was used to plan the development process of MESOC Toolkit.

This thesis is organized as follows: Chapter 2 describes the process of defining the functionalities and creating the final mock-up for the application. Chapter 3 describes the coding phase, and the thesis ends with conclusions and plans for the future work.

2. Creating the mock-up

The mock-up for MESOC Toolkit was created in a tool called Figma. Figma is a vector graphics editor and prototyping tool which is primarily web based. It allows multiple users to work on the same design file at the same time. This offers a way for seamless collaboration between designers, developers and clients. Some of Figma's competitors are:

- Sketch [2]
- Adobe XD [3]
- UXPin [4]

Figma's user interface is considered beginner friendly as it consists of only 7 simple tools to begin with:

1. Move tool,
2. Frame tool,
3. Rectangle tool,
4. Pen tool,
5. Text tool,
6. Hand tool,
7. Comment tool.

Mock-ups consist of two important parts: design elements and user flow diagrams. Firstly, all the design elements needed to stack the mock-up were created using earlier mentioned tools in combination with more advanced and more powerful tools that Figma offers. When creating the design elements, it was necessary to pay attention to the rules of good design and follow the given design language. Secondly, after creating the necessary design elements, they needed to be connected by user flow streams. User flow streams describe the desired behaviour of the application. For example, by connecting a particular button to a given frame element, we are basically saying: "When the user clicks this button, I want the application to switch to the specified screen (frame)". By creating all the elements and connecting them properly, we have successfully created a fully working prototype.

The first prototype version may be finished, but the work on this phase is still far from finished. The first version of the prototype usually requires adjustments and detailed layouts before proceeding with the implementation. After all, this is exactly why it is strongly recommended to create the mock-up before coding since changing the mock-up takes way less time and effort than changing the actual code.

After the intensive collaboration between developers and end-users, the final mock-up is created. After the mock-up definition, the coding phase can be initiated.

3. Coding the application

This phase of the development process was divided into multiple parts. Each part represents one of the application screens, except for the App.js section. App.js section is dedicated to the setup process of the whole application.

3.1.   App.js

The usual starting point of React applications is the App.js file. This is a place where we define the architecture, or in other words, the skeleton of the application [5].

The first, and the most obvious, part of creating the application skeleton is the process of defining the routing of an application. Routing is the ability to move between different parts of an application when a user enters a URL or clicks an element like link, button, image etc [6]. First, it is necessary to know which routes an application will have. The initial routing is defined in the planning phase. Once it is known exactly how the URL structure should look like, it's time to develop the code. React-router is the perfect package for achieving this task. This is a package that provides declarative routing for React. There are five main routes that were declared using react-router's components "Route" and "Switch":

1. /browse,
2. /my-documents,
3. /upload-document,
4. /send-feedback,
5. /:analysisType/:analysisKey.

These routes are reserved for the most important pages of the application. There are couple of more routes like "sign-in" or "verification" that are used for the maintenance pages and user management pages. The code that defines mentioned routes is shown in Figure 1.

```
47      return (
48        <React.Fragment>
49          {appReady ?
50            <div className="App">
51            <Route path='/'>
52              <Navbar userToken={userToken}
53                      setUserToken={setUserToken}
54                      removeAuthCookie={removeAuthCookie}
55                      setUserVerified={setUserVerified} />
56            </Route>
57            <Switch>
58              <Route path="/" exact >
59                <Redirect to="/browse" />
60              </Route>
61              <Route path='/browse'>
62                <Browse />
63              </Route>
64              <Route path='/my-documents'>
65                <MyDocuments userToken={userToken}
66                             userVerified={userVerified} />
67              </Route>
68              <Route path='/send-feedback'>
69                <SendFeedback userToken={userToken}
70                              userVerified={userVerified} />
71              </Route>
72              <Route path='/sign-in'>
73                <SignIn setUserToken={setUserToken}
74                        setAuthCookie={setAuthCookie}
75                        userVerified={userVerified}
76                        setUserVerified={setUserVerified}
77                        verificationUUIDkey={verificationUUIDkey} />
78              </Route>
79              <Route path='/create-account'>
80                <SignUp />
81              </Route>
```

Figure 1: Code that defines routing in MESOC Toolkit

The second major part of an application's setup process is defining the usage of the http cookies. Http cookies, which are also called web cookies or cookies, are small blocks of data created by a web server, and are stored on the user's computer or whichever other device he is using for accessing the page [7]. To create and manage cookies in MESOC Toolkit, the react-cookie package was used. React-cookie is a React specific variant of universal-cookie package [8] which is used to handle cookies universally throughout the whole JavaScript ecosystem. Furthermore, react-cookie provides simple and easy to use hook functions that need to be defined once and can be reused unlimited number of times. A piece of code that defines the usage of user management cookie in MESOC Toolkit is:

```
const    [authCookie,    setAuthCookie,    removeAuthCookie]    =
useCookies(null);
```

Given line initializes three constant variables:

1. authCookie – the cookie value is stored here,
2. setAuthCookie – a function that is used to create or update particular cookie,
3. removeAuthCookie – as the name suggests, a function that is used for clearing a given cookie.

In a similar manner, several more cookies were established and used throughout the application. Those cookies are:

- DGPR Compliant cookie,
- User verification cookie,
- Analysis properties cookie.

This covers all the necessary parts of the App.js file. The next major application part lays in the Browse.js file.

3.2.   Browse.js

The main part of the MESOC Toolkit is located in the Browse.js file. This file consists of an interactive world map that contains a list of all the cities in the MESOC database. The map is shown in the Figure 2. Each city is represented by a clickable pin that, when clicked, displays a popup with brief information about the collection of pilot and scientific studies connected to that city. Each pin leads to the Analysis screen with more in depth analysis of all the studies. The analysis screen will be described in Chapter 3.4.
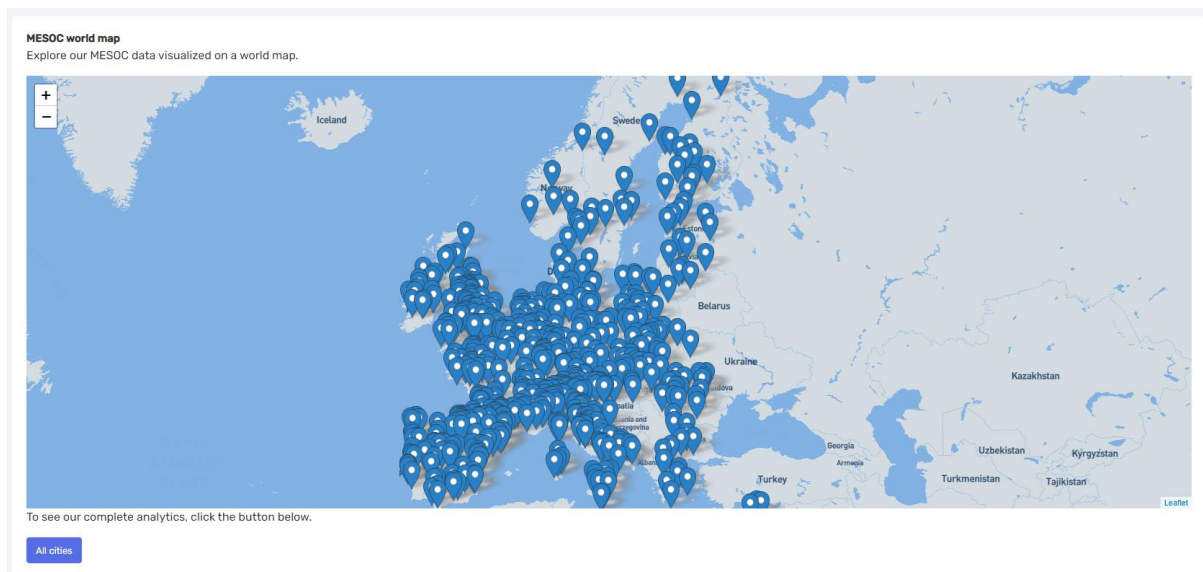

Figure 2: The interactive MESOC world map

The map was created with Leaflet [9] – a JavaScript library for building interactive maps. Leaflet is an open-source library designed with simplicity, performance and usability in mind. This comes with some caveats, for example, the default map and it's settings that it offers out of the box are pretty limited in features. Thankfully, Leaflet can be extended with lots of official, as well as, community made plugins that make up for the default lack of features. One of the plugins that were used for the MESOC Toolkit is mapbox-gl-leaflet. This plugin is a binding from Mapbox GL JS library to a familiar Leaflet library [10]. It offers broader map customization options than the default ones. In this application, Mapbox was used to remove the unnecessary "clutter" from the map. The default map comes with marked roads, institutions, terrain etc. which was not needed for our purposes. The map pins were created in Figma and then imported as png's.

3.3. MyDocuments.js

MyDocuments.js is the first protected page that will be covered in this thesis. The page is intended to be accessible only to registered users and it serves as a file manager. The page is protected with the help of cookies combined with react-router's Redirect component. All files uploaded by the user are displayed here, and are categorized into two main categories:

1. Active documents,
2. Failed documents.

The categorization was made with the help of Reactstrap library [11]. Reactstrap is best described as a library that provides React components enriched with Bootstrap 4 styling. The library does not depend neither on jQuery nor Bootstrap JavaScript, however, popper.js is required for more advanced positioning of content. The categorization was achieved by using tab-based navigation which is shown in the Figure 2.
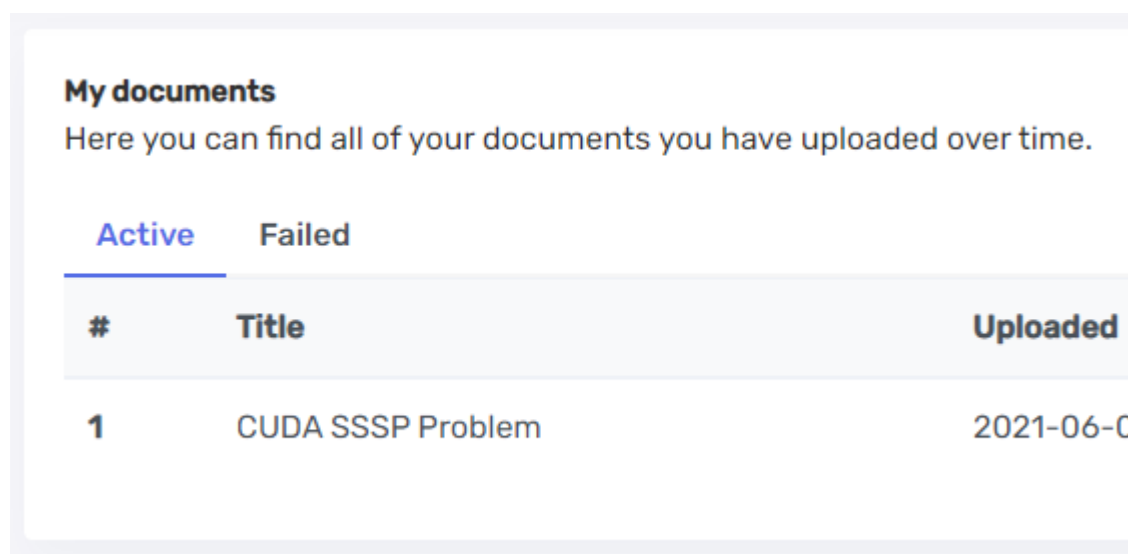

My documents
Here you can find all of your documents you have uploaded over time.

Active	Failed

#	Title	Uploaded
1	CUDA SSSP Problem	2021-06-0

Figure 3: Tab-based navigation in MESOC Toolkit

To get the navigation working, the following set of Reactstrap's components are used: Nav, NavItem, NavLink, TabContent and TabPane [12]. Nav, NavItem and NavLink components were used to create the actual tabs that are clickable and are used for navigating between two groups of documents mentioned earlier. TabContent component was used in combination with TabPane component to create the content placeholders in which our documents would be stored. On top of all that, this page also uses Table component which allows the documents to be easily displayed as a mobile responsive table view.

By clicking on the "open" button of each document, user will be redirected on the analysis of a given document. The analysis is shown on the Analysis screen which is described next.

3.4.   Analysis.js

Analysis.js file is the hearth of MESOC Toolkit. This page comes in a format of business intelligence dashboard and it is responsible for visualisation of the analysis results. This is where users can browse through their desired data in an interactive way. The page consists of three different components:

1. Heatmap,
2. Graph,
3. Two similar document lists.

Heatmap is used for displaying the impact of variables per cell in a given document. A heatmap is a data visualization technique that shows magnitude of a phenomenon as color in two dimensions. In this application, heatmap uses the variation in color intensity along with a percentage to display the impact strengths. There are many heatmap solutions packed as JavaScript libraries or packages but since majority of them is intended for visualization only, it was necessary to create a custom solution for MESOC Toolkit. This heatmap required interactivity in a way that user can select individual cell by clicking on it. Our solution was created using CSS Grid Layout [13]. CSS Grid Layout is a two-dimensional grid-based layout system that aims to allow developers to create responsive user interfaces using grid-like structure. This way, we had a full control over rendered heatmap. The click functionality works thanks to the React state, when user clicks on certain cell, it's index is stored into state so the App can perform further actions based on this information. Figure 3 shows the final result of heatmap rendering.

MESOC matrix

CUDA SSSP Problem, Rijeka, Croatia

	Health and Wellbeing	Urban and Terrotorial Renovation	People's Engagement and Participation
Heritage	0%	0%	0%
Archives	0%	0%	0%
Libraries	0%	0%	0%
Book and Press	0%	0%	0%
Visual Arts	0%	0%	0%
Performing Arts	36%	0%	6%
Audiovisual and Multimedia	22%	0%	4%
Architecture	0%	0%	0%
Advertising	0%	0%	0%
Art crafts	24%	0%	4%

Figure 4: The MESOC matrix represented in a heatmap

The next component is MESOC graph. The graph shows distribution of variables impacting a selected heatmap cell. It was created with the help of Nivo [14]. Nivo is JavaScript library that provides supercharged React chart components built on top of d3 [15] library. The best graph format for this application was bar chart – a chart that presents categorical data with rectangular bars with heights proportional to the values they represent. The usage of Nivo charts is pretty simple so only thing that had to be done before rendering a graph was to sort data in descending order.

Similar document lists are used to display similar documents based on cell similarity or variable similarity. Lists were created using pure CSS since they are relatively simple components compared to the rest of MESOC Toolkit's features.

3.5. UploadDocument.js

The upload document page is used for uploading and submitting new documents to MESOC analysis. The page consists of a simple form which has 4 input fields:

1. Title,
2. Language,
3. Location,
4. File.

Title is a simple html `<input>` element that accepts strings and, as the name suggests, is intended for specifying document title.

Language and Location inputs were a bit more complicated to create. They were made with the help of Reactstrap's `<Select />` component which, under the hood, uses html select tag. Furthermore, since there were a lot of locations and languages to choose from, it was necessary to implement a search functionality. Search works so that when the user enters a string, the application makes a call to a backend server with given string as a parameter. Before making a call, application waits for 60 milliseconds. These two functionalities have allowed only the necessary locations or languages to be retrieved from the server. The 60 millisecond wait has a role to prevent the application from sending too many requests to the server.

The File input, or more precisely, document upload area was created with the help of react-dropzone package. The traditional default way of uploading files that html offers looks too basic, so it was decided to go with more complicated, but also more elegant solution. React-dropzone offers a way to leverage the functionalities of the default `<input type="file" />` element while transforming it into much more appealing drop-zone component. The renderd <FileDropzone /> component used in MESOC Toolkit is shown in the Figure 4.

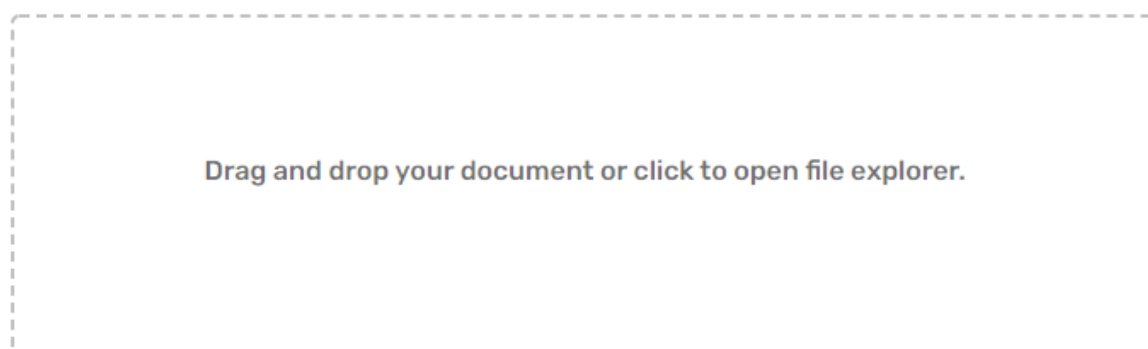Drag and drop your document or click to open file explorer.

Figure 5: Rendered FileDropzone component

The drop-zone can, just like the traditional way, be clicked to open a file explorer or it can be used to "drop" files onto it. Drop-zone like inputs are much better solutions especially for when users have multiple desktops because they make uploading a file more intuitive.

3.6. SendFeedback.js

Lastly, there is a Send feedback page. The important part of the development lifecycle of an application is testing. Testing is done by developers, testers and by users of the application. Developers and testers often do a good job of finding and eliminating most of the technical bugs but there is more to it than just that. Users know the best what are their wants and needs, so it is very important to listen to the application's userbase.

Send feedback page is intended for users to write the feedback on the encountered problems or to provide suggestions for improving the application. From the technical side, this page consists of a simple form with one `<input type="text" />` element for specifying email address, and one `<textarea />` element used for writing the message body.

4. Conclusion

In this thesis, we have discussed the development process of MESOC Toolkit web application. On top of that, there is a general discussion about developing web applications with React JavaScript library. We saw many examples of packages that can be used with React to create robust applications with relative easy. However, React is much more complicated than we have covered in this thesis. Here, we have focused mainly on MESOC Toolkit.

Furthermore, we now have production ready web application that is connected to a backend server that is capable of doing advanced natural language processing.

In the future, it's planned to improve some of the existing application components, and add several more application parts. Two components that will be improved are:

1. MESOC Graph,
2. Document table on /my-documents screen.

The MESOC Graph will be rotated so the bars are spreading horizontally. This will allow long variable names to be fully displayed instead of shortening them to fewer characters. Document table component will receive two improvements. First improvement is formatting the date column to display the date in ISO 8601 standard [16]. The second improvement will be adding sort on click functionality to all column headers.

Components that are planned to be added to MESOC Toolkit are:

1. Top 10 keywords list
2. Variable decomposition screen
3. User confirmation screen

Appendix

Appendix includes a link to a GitHub repository containing a copy of the web application featured in the thesis and the accompanying deployment and configuration instructions.

References

[1] Amanda Athuraliya (last updated: 2021.). Step-by-Step Visual Guide to Mobile App Planning. Retrieved from https://creately.com/blog/diagrams/how-to-plan-an-app-visually/

[2] Official Sketch website. Retrieved from https://www.sketch.com/

[3] Official Adobe XD website. Retrieved from https://www.adobe.com/products/xd.html

[4] Official UXPin website. Retrieved from https://www.uxpin.com/

[5] Adam Boduch (2020.) React and React Native: A complete hands-on guide to modern web and mobile development with React.js, 3rd Edition

[6] Gaurav Singhal (2020.). Pros and Cons of Client-side Routing with React. Retrieved from https://www.pluralsight.com/guides/pros-and-cons-of-client-side-routing-with-react

[7] Web technology for developers – Using HTTP cookies (2021.). Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

[8] GitHub repository and official documentation of universal-cookie package. Retrieved from https://github.com/reactivestack/cookies/tree/master/packages/universal-cookie

[9] Official Leaflet website and documentation. Retrieved from https://leafletjs.com/

[10] Official Mapbox GL JS website and documentation. Retrieved from https://docs.mapbox.com/mapbox-gl-js/api/

[11] Official reactstrap website and documentation. Retrieved from https://reactstrap.github.io/

[12] List and detailed description of all reactstrap's components. Retrieved from https://reactstrap.github.io/components

[13] Web technology for developers – CSS Grid Layout (2021.). Retrieved from https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout

[14] Official Nivo website and documentation. Retrieved from https://nivo.rocks/

[15] Official d3 website and documentation. Retrieved from https://d3js.org/

[16] Official ISO Standards website. Retrieved from https://www.iso.org/iso-8601-date-and-time-format.html

Table of figures