BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF BUSINESS AND MANAGEMENT

FAKULTA PODNIKATELSKÁ

INSTITUTE OF INFORMATICS

ÚSTAV INFORMATIKY

MODERN ANALYSIS METHODS OF POLITOLOGY DATA USING JAVASCRIPT AND NOSQL DATABASE

MODERNÍ METODY ANALÝZY POLITICKÝCH DAT POMOCÍ JAZYKA JAVASCRIPT A NOSQL DATABÁZE

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR

AUTOR PRÁCE

Aleksandr Voznesenskii

SUPERVISOR

VEDOUCÍ PRÁCE

Ing. Jiří Kříž, Ph.D.

BRNO 2022

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF BUSINESS AND MANAGEMENT

Assignment Bachelor's Thesis

Department:	Institute of Informatics
Student:	**Aleksandr Voznesenskii**
Supervisor:	**Ing. Jiří Kříž, Ph.D.**
Academic year:	2021/22
Study programme:	System Engineering and Informatics
Study field:	Managerial Informatics

Pursuant to Act no. 111/1998 Coll. concerning universities as amended and to the BUT Study Rules, the degree branch supervisor has assigned to you a Bachelor's Thesis entitled:

Modern Analysis Methods of Politology Data Using JavaScript and NoSQL Database

Characteristics of thesis dilemmas:

Introduction
Objectives of the work, methods and procedures of processing
Theoretical basis of the work
Analysis of the current state
Custom solutions
Conclusion
List of used literature
Side dishes

Objectives which should be achieve:

The aim of this work is to design an information portal for research and analysis of political data.

Basic sources of information:

GOLOSOV, Grigorii V. The Effective Number of Parties. Party Politics. 2009, 16(2), s. 171-192. ISSN 1354-0688. Dostupné z: doi:10.1177/1354068809339538

GOLOSOV, Grigorii V. Komparativní politologie. 1. vyd. Sankt-Peterburg: EUPRESS, 2018. ISBN 978-5-94380-211-9.

Gatsbyjs.com The fastest frontend for the modern web © 2021 Gatsby, Inc.Dostupné z: https://www.gatsbyjs.com/

GraphQL.org A query language for your API © 2021 The GraphQL Foundation.Dostupné z: https://graphql.org/

Deadline for submission Bachelor's Thesis is given by the Schedule of the Academic year 2021/22

In Brno dated 28.2.2022

L. S.

doc. Ing. Miloš Koch, CSc. Branch supervisor	doc. Ing. Vojtěch Bartoš, Ph.D. Dean

Abstract

Volební historie je mezi politology oblíbeným tématem výzkumu. Přestože je mnoho údajů o jednotlivých zemích snadno dostupných, žádný z datových souborů je nespojuje do databáze s otevřeným přístupem. Tento úkol je klíčový pro srovnávací politologický výzkum. Moje práce má za cíl vytvořit otevřenou službu, která by umožnila efektivnější a rychlejší zpracování volebních dat.

Electoral history is a popular research topic among political scientists. While much country-specific data is readily available, none of the datasets combine them into a database with open access. This task is crucial for comparative political science research. My thesis strives to create an open service that would allow for processing electoral data more efficiently and faster.

Keywords

Vizualizace dat, Uživatelské rozhraní, JavaScript, Politologie.

User Interface, JavaScript, Data visualization, Political science.

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne
16. dubna 2022

.....
podpis autora

Acknowledgements

At the beginning of this work I would like to thank **Vladimir Kudryavtsev**[1], researcher at the Department of Political Science of the European University of St. Petersburg[2] who had the most significant influence on this project. He helped shape the vision and had also provided research materials.
I would also like to thank my colleauges from **Red Hat**
Marie Doruskova[3] who consulted me on UX requirements and helped me to create a better understanding of the end-user.
Georgy Karataev[4] for his reviews and comments on how to improve my thesis.
Chris Santiago[5] for his ideas and help with development of prototype.
Pavel Tisnovsky[6]for sharing his engineering experience and consultation.
Denis Goncearenko[7] for his help with thesis review.
Olga Kholkovskaia for her help with editing the thesis.
My friends and family supported me during my study at the university.

[1] Vladimir Kudryavtsev
[2] Department of Political Science of the European University of St. Petersburg
[3] Marie Doruskova
[4] Georgy Karataev
[5] Chris Santiago
[6] Pavel Tisnovsky
[7] Denis Goncearenko

Contents

Rozšířený abstrakt

Motivace

Informační technologie a data přinášejí změny do všech oblastí našeho života. Data se hromadí exponenciální rychlostí, což přináší nové výzvy jak pro počítačová, tak pro sociální médiá a vědy. Jednou z nich je poskytnout běžnému člověku spolehlivé informace o volbách, které by bylo možné provést pouze ve spolupráci technických a sociálních odborníků. Je to důležitý věc, na které bychom měli celek jako společnost pracovat, abychom zabránili manipulacím s daty, které mohou ovlivnit preference voličů a ovlivnit výsledky voleb. Cílem teto práce je vytvoření nástroje, který pomůže lidem učit se, vytvářet a šířit správné, ověřené a objektivní informací týkajících se voleb.

Současný stav

V současné době lze na internetu nalézt mnoho webových stránek s nejrůznějšími údaji o volbách. Různé volební databáze poskytují informace na regionální, národní i mezinárodní úrovni. Volební kalkulačky počítají různé ukazatele a slouží jako doporučující systém, který uživateli najde nejvhodnějšího kandidáta. Ne všechny takové systémy jsou otevřený a poskytují přístup k surovým datům a zdrojovému kódu. To by mohlo způsobit, že jejich výsledky nebudou ověřitelné. Příkladem spolehlivých zdrojů politických dat jsou ”Election database”[8] a ”Freedom House”[9].
”Election database” uchovává podrobné výsledky parlamentních voleb z celého světa a poskytuje je zdarma. Nenabízí však žádné ”rychlé” výpočty nebo služby, nejedná se o webovou aplikaci. ”Election database” poskytují pouze data ke stažení v různých formátech, se kterými je obecně těžkopádné pracovat.
”Freedom House” vydává každoročně zprávy o stavu svobody a politických trendech od 50. let 20. století. Jejich metodika vychází ze Všeobecné deklarace lidských práv a je k dispozici na webových stránkách organizace. Výsledky několika ukazatelů, jako je globální svoboda, svoboda internetu a demokracie, jsou prezentovány ve webové aplikaci, která vizualizuje aktuální stav ukazatelů a trendy v jednotlivých zemích. Na druhou stranu má jednoduchý přístup k nezpracovaným údajům a omezený počet metrik.
Cílem této práce je vytvořit open-source projekt, který by nejen uchovával spolehlivá celosvětová volební data, ale také poskytoval jednoduchý uživatelský přístup k surovým datům v kombinaci s několika důležitými metrikami pomocí vědecky ověřených metod. Taková služba by mohla zlepšit přístup zkušených odborníků a politologů k datům. Kromě toho by byla přínosná i pro běžného člověka, neboť by mu umožnila budovat si představu o současné politické situaci a politické historii v jeho zemi.

[8] Election database
[9] Freedom House

Popis projektu

Navrhovaný projekt se skládá ze 3 hlavních částí: Výzkum uživatelských zkušeností, webové stránky se základním přístupem k datům a automaticky vypočítané metriky.

Uživatelské zkušenosti

Prvním krokem je **Výzkum uživatelských zkušeností (UX)**, studie cílových uživatelů, která pomůže shromáždit a analyzovat informace pro proces vývoje produktu. Porozumění potřebám koncových uživatelů, způsobu jejich interakce s daným produktem, získá cenné poznatky o jejich chování a je klíčové pro každý projekt. Tyto potřeby bychom měli identifikovat, abychom mohli jasně formulovat požadavky na produkt a připravit návrh, který by mohl být použit pro implementaci. Zpracováním uživatelských zkušeností, konkrétními metodami výzkumu uživatelů a návrhem produktu se zabývá kapitola 2.

Vývoj webových stránek

V dalším kroku představíme webové stránky se základními funkcemi. Měl by poskytovat otevřený přístup k databázi obsahující historii voleb podle zemí připravenou k analýze. Uživatelé by měli mít možnost si stáhnout nezpracované údaje pro vybranou zemi v jednoduchém známém formátu, jako jsou hodnoty oddělené čárkou (CSV). Volba **React** 1.2 svědčila o tom, že hmatatelné User Interface 4.1 lze vytvořit zcela v mezích **"světu JavaScript"**. Knihovna **PatternFly** 1.6.1 poskytuje přístup k rozsáhlé škále nástrojů a komponentům nezbytným pro neabstraktní zajištění pozitivního uživatelského zážitku a zjednodušení vývoje User Interface 4.1. Další podrobnosti o vybraném technologickém zásobníku a samotném procesu implementace jsou k dispozici v kapitole 3.

Kvalita životních funkcí

Posledním krokem je implementace řady funkcí v podobě automaticky vypočítaných metrik. Mezi vybrané metriky patří počet politických stran, počet křesel získaných každou stranou v parlamentu a **Efektivní počet stran**, což je míra fragmentace politického systému. Výběrem metrik a výpočetními metodami se zabývá kapitola 1.

Propagace a budoucí vývoj

Po dokončení vývoje **Minimal Viable Product (MVP)** 4.1 by měl být projekt propagován v akademických a IT kruzích mezi lidmi, kteří mají zájem přispívat a dále jej vylepšovat. To by umožnilo připravit pevné základy pro jeho budoucí vývoj jako open-source projektu. Možným dalším krokem by mohl být seznam s požadavky na nové funkce. Dalším důležitým zlepšením je vývoj aplikačního rozhraní (API). To posune projekt na novou

úroveň a zvýší pravděpodobnost integrace třetích stran a komerčního úspěchu. Budoucí vývoj a možná vylepšení jsou popsána v kapitole 4.1.

Výsledky a diskuse

V rámci této práce jsme prokázali, že vývoj navrhované webové aplikace je možný. Ve dnešní době úložiště pravdivých, nezávislých zdrojů dat o historii voleb, které by bylo převedeno do interaktivního média přístupného průměrnému laikovi v jednoduché a srozumitelné formě, by významně zlepšilo schopnost zmíněného laika orientovat se v moderním politickém prostředí. Po zdokonalení služby a jejím naplnění daty ji budou ji moci využívat politologové a novináři jako **důvěryhodný a objektivní zdroj** pro své články a reportáže. Projekt by mohl sloužit ke zlepšení úrovně současného politického vzdělání, který průměrný člověk dostává. To by jim umožnilo vytvořit si nezávislý názor a činit lepší a informovanější politická rozhodnutí. Služba by umožnila sledovat vývojové trendy v oblasti politických studií. Open-source charakter projektů umožňuje širokému okruhu odborníků se stát přispěvateli a zapojit se do vývoje.

Další vylepšení

Během vývoje MVP 4.1 bylo zjištěno několik slabých míst v procesu vývoje, která by měla být vylepšena. Hlavními problémy současné implementace jsou:

1. Obrovský objem údajů o historii voleb vedoudci k přílišné složitosti datového schématu. Abych lépe pochopil, jak to vyřešit, konzultoval jsem to s **Red Hat** inženýrem **Pavlem Tisnovským**[10], který navrhl schůdné řešení uvedeného problému pomocí databázového systému PostgreSQL 4.1.

2. Použití knihovny **PatternFly** 1.6.1 je odůvodněno malým rozsahem MVP 4.1 a přináší omezení. Nástroje pro vizualizaci dat nabízejí snadné a rychlé možnosti prototypování, ale jejich nepružnost ve srovnání s knihovnou **D3** je činí nedostatečnými mimo rámec MVP 4.1. Knihovna **D3** by byla nezbytná pro dosažení vývojových cílů v budoucnu.

Hlavním omezením pro budoucí rozvoj tohoto projektu by byla schopnost zajistit potřebné finanční prostředky. Zůstat zcela otevřený a nezávislý a zároveň pevně deklarovat záměr. Záměrem není poučovat, ale informovat. Toho by bylo možné dosáhnout zajištěním dalších finančních prostředků od oficiálních institucí, jako je Ministerstvo Školství. Dlouhodobě robustnějším přístupem je však zorganizování charitativní organizace nebo zřízení systému dárcovství.

[10] Pavel Tisnovsky

Extended Abstract

Motivation

Information technologies and data bring changes to all sides of our lives. The data is being accumulated at an exponential rate, bringing new challenges both to computer and social sciences. One of them is to provide a regular person with reliable electoral information that could be only done in cooperation between technical and social experts. It is an important thing that we should work on together as a society to prevent data manipulations that can sway voters' preferences and influence election results. This thesis strives to create a tool that will help people to learn, create and spread the correct, verified and unbiased election-related information.

State of the Art

Nowadays numerous web-pages with all kind of election data could be found on the web. Different election databases provide information on regional, national, and international level. Election calculators compute various metrics and serve as a recommendation system finding the most suitable candidate for the user. Not all such systems are open-source, and luck of access to raw data and source code could makes their results unverifiable.
An example of reliable sources for political data are the election database [11] and Freedom House [12].
The election database stores detailed election results for legislative elections from around the world, and makes it available at no cost. But they do not offer any "quick" calculations or services, it is not a web application. Election database only provide downloadable data in different formats that are generally cumbersome to work with.
Freedom House issues yearly reports on freedom status and political trends from 1950s. Their methodology is based on the Universal Declaration of Human Rights and is available on the organization's web-site. Results for several metrics, such as global freedom, internet freedom, and democracy are presented in the web application that visualizes the current state of the metric and trends country-wise. On the hand, it lucks access to the raw data and has limited number of metrics.
The goal of this work is to create a open-source project that would not only store reliable worldwide electoral data but also provide simple user access to the raw data combined with several important metrics calculated with scientifically proven methods. Such a service could improve the way experienced professionals and political scientists access the data. Furthermore, it would be beneficial to a regular person as it would enable them to build up their understanding of the current political situation and the political history in their country.

[11] Election database
[12] Freedom House

Project Description

The proposed project consists of 3 major parts: User Experience research, website with basic access to data, and automatically calculated metrics.

User Experience

The first step is **User Experience (UX)** research, the study of target users that will help to collect and analyze information for product development process. Understanding end-user needs, the way they interacts with your product gains valuable insights of their behaviour and is crucial to any project. We should identify those needs to clearly formulate product requirements and prepare a design that could be used for the implementation. User experience process, specific user research methods, and product design are discussed in Chapter 2.

Website Development

As the next step, we are going to introduce a website with the basic functionality. It should provide open access to the database containing analysis-ready electoral history by country. Users should have an opportunity to download raw data for the selected country in the simple well-known format, like comma-separated values (CSV). The choice of **React** 1.2 had been a testimony to the idea that a tangible User Interface 4.1 could be built completely withing bounds of the **"JavaScript realm"**. The **PatternFly** 1.6.1 library gives access to a vast range of tools and components necessary for the unabstracted provision of positive user experience and simplification of the User Interface 4.1 development. More details on the selected technology stack and implementation process itself are available in Chapter 3.

Quality of Life Features

The last step is to implement a number of **quality of life** features in the form of the automatically calculated metrics. The selected metrics include numbers of political parties, the number of seats obtained by each party in a parliament, and **Effective Party Number**, a measure of party system fragmentation. The metric selection and the computational methods are covered in Chapter 1.

Promotion and Future Development

After the **MVP** 4.1 development is finished, the project would be promoted in the academic and IT circles among people interested in contributing and further improving it to prepare a strong baseline for its future development as an open-source project. Possible next steps could be a list with new feature requirements. Another important improvement is the development of an Application **Application Programming Interface (API)**. That will

that will push the project to a new level and increase the likelihood of the third-parties integration and commercial success. Future development and possible improvements are discussed in Chapter 4.1.

Results and Discussion

Within this thesis, we have proven that developing a proposed web application is possible. Nowadays, a repository of truthful, independent sources of electoral history data made into an interactive medium accessible to the average layperson in a simple and understandable form would seriously improve the ability of the said layperson to orient themselves within the modern political landscape. After the service is improved and filled with data, it shall be used by political science researchers and journalists to provide a **credible and unbiased source** for their articles and reports. The project could be used to improve the level of current political education received by an average person. That would allow them to form an independent opinion and make better and more informed political decisions. The service would allow to monitor the developing trends in the field of political studies. The open-source nature of the projects makes it possible to wide range of professionals to join the development and become the contributors

Further Improvement

Throughout the development of the MVP 4.1, I have identified several weak spots in the development process that should be improved. The main problems of the current implementation are:

1. The vast volume of the electoral history data leads to the excessive complexity of the data schema. To better understand how to solve this, I had been consulting with a **Red Hat** engineer **Pavel Tisnovsky**[13] who proposed a passable solution to the said problem through the use of a database system called PostgreSQL 4.1.
2. The use of the **PatternFly** 1.6.1 library is warranted by the small scope of the MVP 4.1 and comes with its own limitations. Data visualization instruments offer easy and quick prototyping possibilities, but their inflexibility compared to the **D3 library** makes them inadequate outside the boundaries of MVP 4.1. The **D3 library** would be necessary for the achievement of the developmental goals down the line.

The main **limiting bottleneck** for the future development of this project would be the ability to secure necessary funds while firmly stating the intention to remain completely open and unbiased. The intent is not to instruct but to inform. That could achieve by securing additional funding from official institutions, like the Ministry of Study. But a more robust long-term approach is to organise a charity or set up a donations system.

[13] Pavel Tisnovsky

Chapter 1

Theoretical basis

1.1 Web Application Architecture

In 2007, **Jeff Atwood** made the quote that was popularly referred to as **Atwood's Law**: "Any application that can be written in JavaScript, will eventually be written in JavaScript."

In this chapter, I am elaborating on why JavaScript 4.1 and its superset TypeScript 1.1.5 were chosen as programming languages to develop this application. It covers the technology stack, libraries, and frameworks and explains how they work together.

1.1.1 How Browsers work

To better understand how to create a modern web application, it is crucial to know how it is built. The modern Internet is based on a solid foundation of the web standards - technologies we use to build websites. They are created by standards bodies institutions (like **World Wide Web Consortium**) that gather specialists from different technology companies. Such organizations create technical specifications, which detail how the technology should work.

1.1.2 JavaScript engines

V8 is Google's open-source **JavaScript** and **WebAssembly** engine, written in C++. It is used in Chrome and Node.js and other programs. Companies like Mozilla and Apple create engines such as SpiderMonkey or JavaScriptCore for their browsers. [1]

[1] *v8 Engine documentation* 2021 *Node.js documentation describing browser engines* 2021

1.1.3 Current state of JavaScript

1. With each year, JavaScript and the V8 engine work better than before. The open-source V8 engine developed by "The Chromium Project" aimed at producing faster, more stable technology to support the entire Web.[2]

2. JavaScript and its application framework/library can be used on the website (client) side and, with the help of Node.js JavaScript runtime, on a server. [3]

3. A JavaScript framework, Electron, enables programmers to write native desktop applications using the knowledge of JavaScript language specifics and syntax. Electron combines the Chromium and Node.js rendering engines. [4]

4. Node.js is designed for writing highly scalable web applications. Node Package Manager (NPM), is the most popular JavaScript package manager globally. NPM contains more than 1.3 million software packages.

1.1.4 ECMAScript Standard

ECMAScript is the scripting language JavaScript 4.1 and TypeScript based on. The official document describing the language specification and processes is described on the official website of **TC39 - Technical Community 39**. The TC39 committee is responsible for iterating and evolving the **ECMAScript** language specification. [5]

After the default step of proposal creation, the **TC39** uses a 4 step process to review language proposals. As a result of this process they publish new ECMAScript editions 4.1.

1. Make a case for the addition, describe the shape of a solution, and identify potential challenges.

2. Describe the syntax and semantics using formal specification language.

3. Indicate whether further refinement will require feedback from implementations and users.

4. Indicate that the feature is ready for an introduction to the traditional **ECMAScript** standard.

[2] Keizer 2020
[3] Brewster 2021
[4] *Introduction to Electron* 2021
[5] *ECMAScript Language Specification* 2022

1.1.5 TypeScript

Developed by **Microsoft**, **TypeScript** is the second most used programming language based on the **ECMAScript** language specification [6]. It is a strict syntactical superset of JavaScript and adds static typization to the language. Browsers cannot natively interpret **TypeScript**. Hence, if the program code is written in **TypeScript**, it must be compiled and converted into JavaScript. The language popularity among developers grows each year since it brings considerable advantages in the development of web applications: [7]

1. **TypeScript** always points out the type errors before the code is compiled into JavaScript.
2. Predictability allows for preventing data mutation. It enhances the likelihood of functions working the way that was initially intended.
3. In the end, **TypeScript** transforms into the same JavaScript that uses the same syntax and could be compiled into the older edition of ECMAScript to be understood by any browser.
4. Integrated development environments (IDE) support provides information about types, making editors and IDE much more helpful. They can offer features like auto-completion, code navigation, and providing accurate suggestions.

The **PatternFly** 1.6.1 library documentation provides a template to create a user interface 4.1 in **TypeScript**, so I will use it to create my prototype.

1.2 What is React

The **React** library helps to create JavaScript 4.1 components for a website to present user data. **React** is one of the most popular and sought-after libraries with a large community. **React**'s main GitHub 4.1 repository has more than 185 thousand "stars," making it the 8th most liked repository globally. **React** allows us to have only one HTML 4.1 file for the whole website and change and quickly display the required data according to users' actions. For this reason, the website does not request new HTML, CSS 4.1, and JavaScript files that allow you to switch between "pages" almost instantly.

1.2.1 Multiple-page applications (MPA)

Multiple-page application (MPA) 4.1 consist of several (more than 1) HTML 4.1 files and have several issues which are absent when you turn to Single-page application (SPA) 4.1.

[6] *ECMAScript Language Specification* 2022
[7] Tate 2015 *The Good and the Bad of TypeScript* n.d.

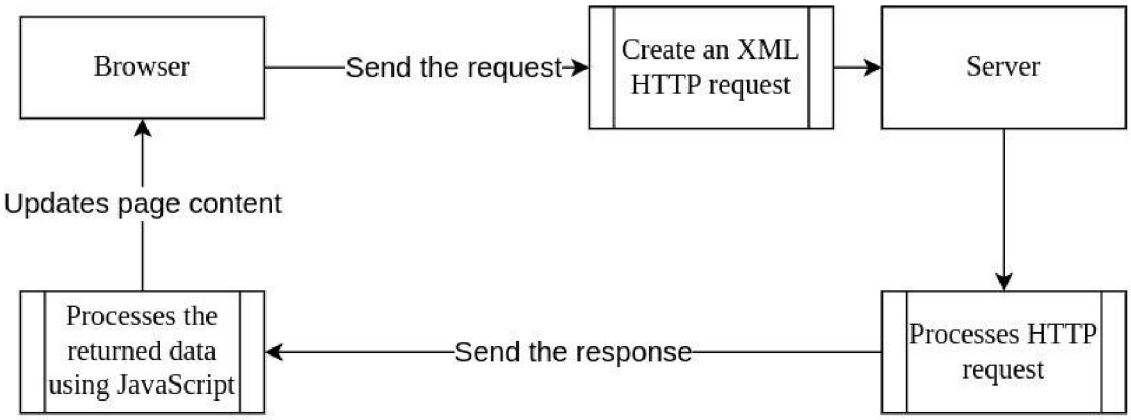

Figure 1.1: Multiple-page application data flow visualization

1. In the case of many requests, the performance issues are highly likely and could cause the speed drop.
2. It takes more time to develop and test because the front end to back end integration is more complex
3. It's harder to maintain and provide updates on a large number of HTML 4.1 pages at the same time

As you can see in Figure 1.1 MPA 4.1 work using HTTP 4.1 Post and Get requests to update the data in the database or receive the HTML 4.1 page. If the user opens a new HTML 4.1 page, the entire **Document Object Model** 4.1 should be unmounted and mounted again with the new data, which increases the load time.

1.2.2 Single-page Application (SPA)

As you can see in Figure 1.2, the HTML 4.1 file loads only once. Following user actions create AJAX 4.1 requests, and server responds by sending JSON 4.1 objects. [8]
This is what the React documentation tells us about SPA 4.1 applications:
"A SPA 4.1 is an application that loads a single HTML 4.1 page and all the necessary assets (such as JavaScript 4.1 and CSS 4.1) required for the application to run. Any interactions with the page or subsequent pages do not require a round trip to the server, which means the page is not reloaded." (*Glossary of React Terms* 2022)

Libraries like **React Router** allow for wrapping the entire application in the **"Browser-Router"** component and instructing it to render a specific **React** 1.2 component when the URL changes. Figures 1.3 and 1.4 visualize how **React** router interacts with the web browser by dynamically rewriting the current web page with the new data from the web server and does not require reloading.

[8] *Single-Page Application vs Multi-Page Application: Pros, Cons, and Which is Better?* 2020

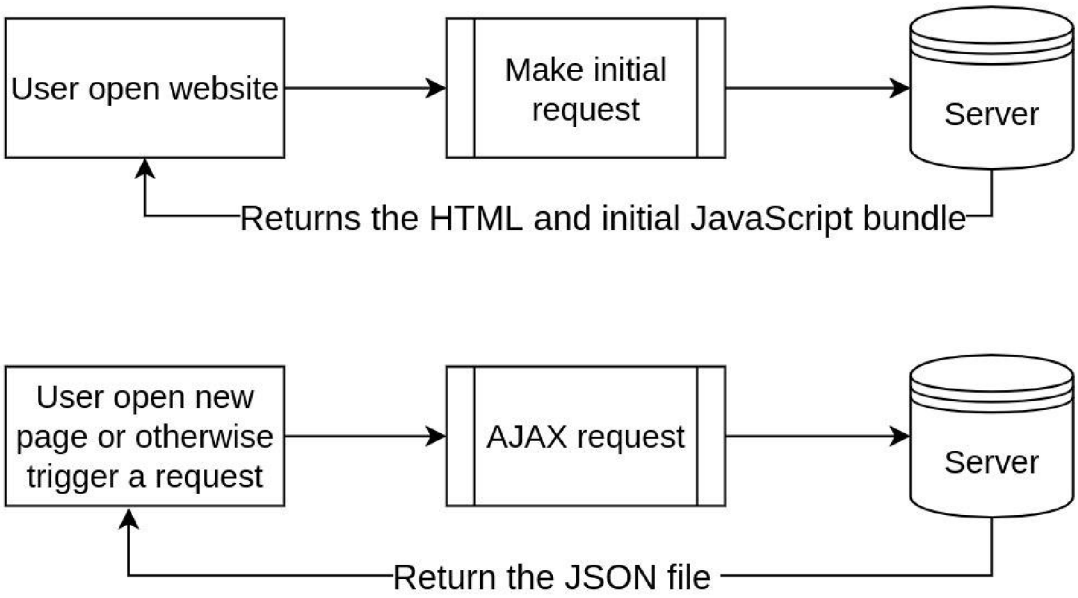

Figure 1.2: SPA website data flow visualization

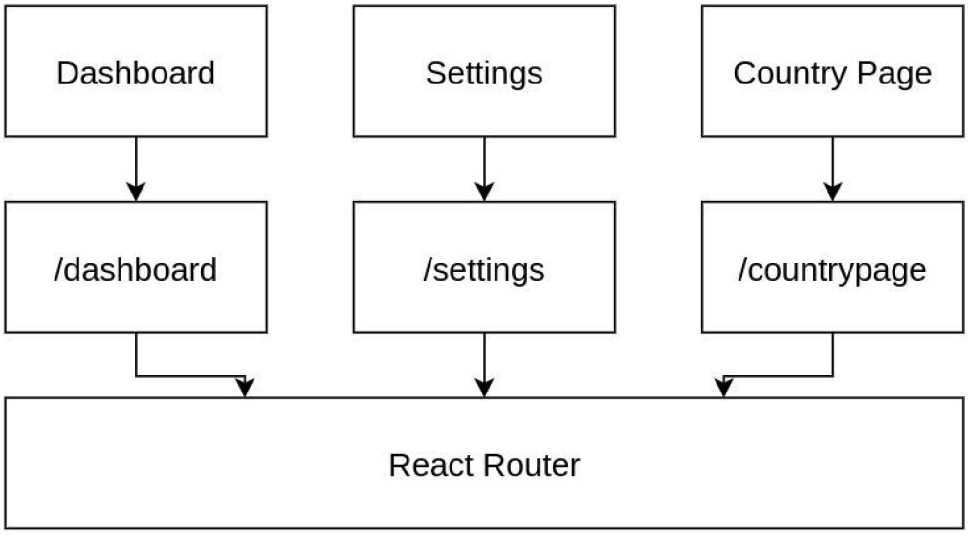

Figure 1.3: React Router component diagram

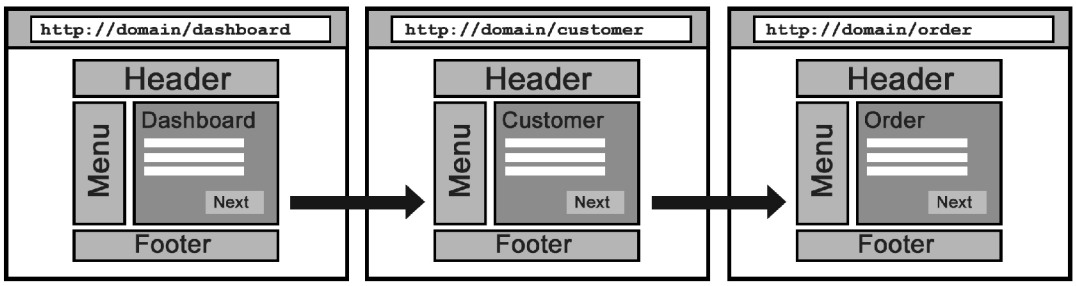

Figure 1.4: DOM 4.1 elements update schema. E.g. when user clicks on the "Customer" button React render only the necessary components to update the page

1.3 What is Node Package Manager

To understand **React**, we need to understand the modern JavaScript 4.1 ecosystem better and why it is that important. Developers use NPM 4.1 to add and update packages and libraries to their projects. If one wants to build an application using **React**, **Vue**, or **Angular**, NPM 4.1 will eventually become a part of the technology stack. I used NPM 4.1 to install and update libraries that provide methods and functions in this thesis.

1.3.1 React Components

The React.js library allows you to split a web page into components 4.1 to further separate business logic. The components 4.1 can communicate with each other and store their data (enabled by the concept of "states"). The **React** components 4.1 are created using a special **JSX** syntax following a specific template construction. [9] Figure 1.5 shows an example of basic **JSX** component.

JSX

```
1  import React from 'react';
2
3  export const Example = () => {
4  return (
5    <div>
6       <p>
7         Example paragraph
8       </p>
9    </div>
10  )
11 }
```

Figure 1.5: Basic JSX component

[9] *Components and Props* 2022

```
import React from 'react';
import { Example } from './example';

export const BiggerExample = () => {
 return (
   <div>
    <Example />
    <Example />
    <Example />
    <Example />
   </div>
  )
}
```

Figure 1.6: We imported the Example functional component into our component called BiggerExample (a functional component) and added 4 Example components into the return statement. That way, we could create a Nested component structure for our application that is easy to duplicate, debug and change.

The most straightforward and recently widely accepted way to create a **React** component 4.1 is to declare a so-called "functional component 4.1". As you can see in Figure 1.6, we created a functional component 4.1 called Example using the arrow function method. Curly braces are allowing us to use JavaScript and call a return that gives us back an HTML 4.1 tag with an <p> inside of it.

Why JSX is a better approach than regular HTML and JavaScript?

1. We can easily use both HTML 4.1 and JavaScript in one file without creating a <script> tag.

2. The React uses a Babel transpiler that transforms **JSX** syntax into the ES5 JavaScript that any current browser could read without any exceptions or errors.

3. It is easy to import the component 4.1 anywhere in your application and create any amount of them without repeating the same code repeatedly. In other words - **JSX** syntax allows us to create a Dryer (**Do not repeat yourself**) codebase.

4. It allows updating **Document Object Model** 4.1 elements easier. With **JSX**, there is little reason to use document.querySelector or other direct **Document Object Model** 4.1 API 4.1 methods.

1.4 How to pass data in React application

1.4.1 Using Properties

```
1   import React from 'react';
2
3   const Hello = (props) => {
4    return (
5     <div>
6      <h1>
7       Hello, {props.name}!
8      </h1>
9     </div>
10   )
11  }
```

Figure 1.7: Component properties example

Components usually need to have a means of interchanging data between each other. In the **React** ecosystem, this is solved by using "properties." They can encapsulate and pass functions, objects, and other primitive values.
Figure 1.7 provides a simple show-case of how the properties can be used in a component 4.1 created with a JavaScript 4.1 function (functional component 4.1).
Once, having the **Hello** component 4.1 declared in this manner, calling **<Hello name="Alex" />** will result in rendering the header containing the text **"Hello, Alex."**

The concept of properties provides a way of sending any data to the "children components 4.1" of any nesting level. For example, in Figure 1.8, it is illustrated how data imported in one of the components 4.1 from a JSON 4.1 file can be subsequently passed to its children components 4.1 where the data is being finally processed or rendered. [10]

[10] *Components and Props* 2022

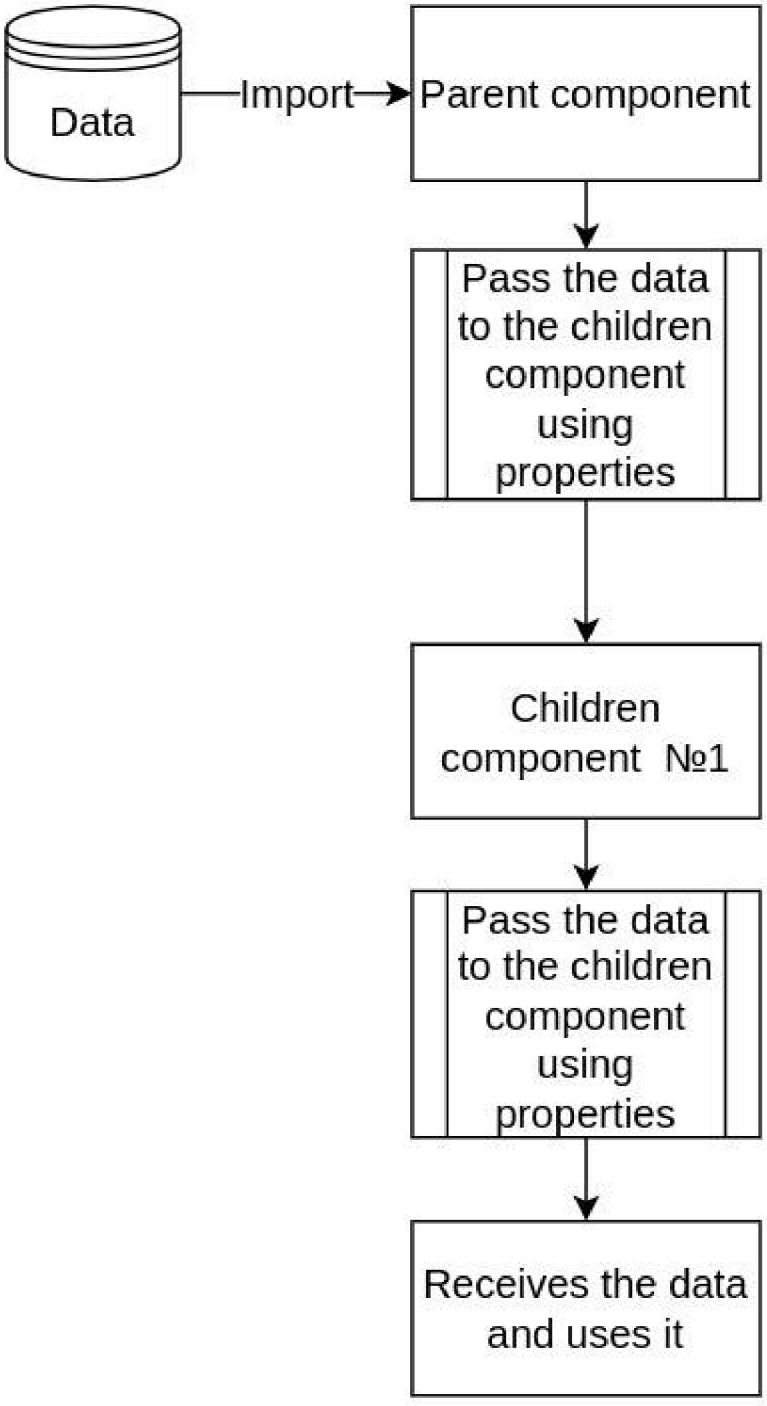

Figure 1.8: Data Flow diagram of component tree passing data using component properties

1.4.2 Using React Context

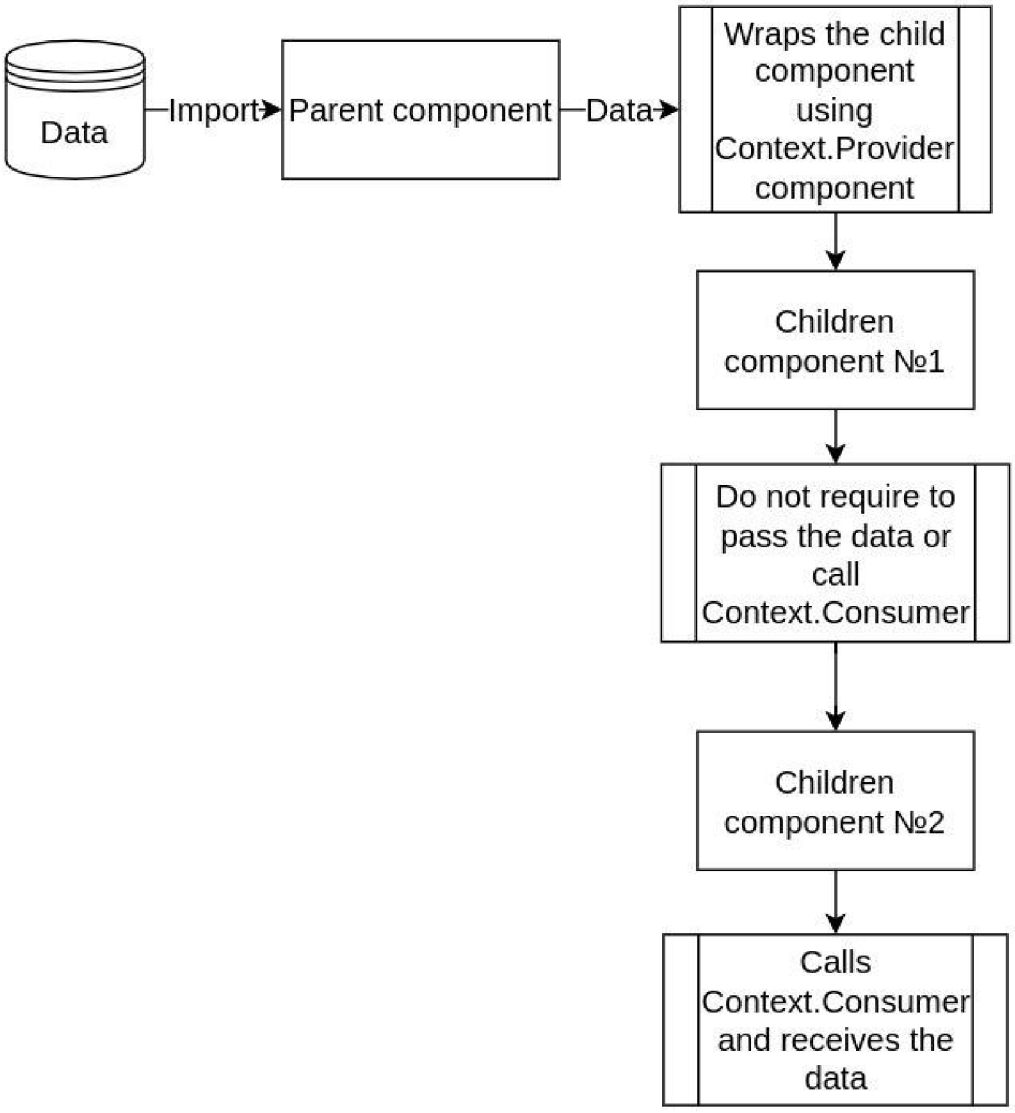

Figure 1.9: Data Flow diagram of component tree passing data using React Context

React Context[11] is designed to share data across the component 4.1 tree. It is an API built into the **React** library starting from version 16. It eliminates the most significant issue of passing props – props "drilling". [12] **Context** is best manifested when the data is "static" and should be accessible by multiple components at any level of the component 4.1 tree. [13]

The data flow between parent and children component 4.1 is depicted in the Figure 1.9. On a small application scale, the issues of using properties are negligible, and, hence, they have been chosen in developing the project's MVP 4.1.

[11] React Context
[12] Doods 2021
[13] *Context* 2022

1.5 State management in React application

```
import React from 'react';

class Clock extends React.Component {
 constructor(props){
  super(props);
  this.state = {date: new Date()};
 }

  render() {
   return (
    <div>
     <h1>
      it is {this.state.date}.
     </h1>
    </div>
   )
  }
}
```

Figure 1.10: React Class component

1.5.1 Using local state

There are two common ways to store data locally in a component 4.1. The first is the class constructor method which can be called after declaring a class component 4.1 (instead of a new style of functional components). In the Figure 1.10, we create a class component 4.1 Clock with a local state consisting of an object with the only property date. This property encloses an instance of the object Date created by calling the object's constructor method. Then could render this component 4.1 onto the component 4.1 in the application, and it will return a div with the header that shows the output of the **Date() function**. The first is the class constructor method that could be called after declaring a new Class. It allows us to declare the different parts of the state.

1.5.2 React Hooks

ES6 classes, unlike functional components 4.1, are not compatible with **React Hooks** (a handy and popular addition to the React library). The majority of React developers prefer to use **React Hooks** because they are easy to understand and take less code to write. **React Hooks** were added in **React** version 16.8. One of them, is that useState is the most popular way to control local component 4.1 state.
The Figure 1.11 show how hooks enabling stateful logic without changing the component 4.1 hierarchy. One of the significant **React Hooks** features is that they let you split one component 4.1 into smaller functions based on the developer's needs. This is an example from **React** 1.2 documentation of how you could set up an easy button with a counter.

```
import React, { useState } from 'react';

export const Example2 = () => {
 const [count, setCount] = useState(0)

 return(
  <div>
   <p>
    You clicked {count} times
   </p>
   <button onClick={() => setCount(count+1)}>
    Add 1 to the count
   </button>
  </div>
 )
}
```

Figure 1.11: React Hooks counter example

React Hooks are convenient to use since they allow setting up specific triggers depending on the state or properties update or a component 4.1 re-render.

1.5.3 Global state

Several state management libraries provide a so-called global state to the whole application. Management libraries move business logic outside the interface and provide the one source of truth to the whole application.[14]
The popularity of libraries like **Redux** and was caused by the increasing complexity of the web pages and web applications. Our codebase must manage more states than ever before since it could include, for example, server responses, cached data, or locally created data that has not yet been sent to a server.

User Interface 4.1 state also increases complexity, as an application has to manage, for instance, active routes, selected tabs, spinners, pagination controls, or color themes. **Redux** provides a read-only state rather than the mutable state. Only a pure function called ”reducers” can update the state after a component 4.1 has sent an action through a dispatcher. While The downsides to having a global state do not disappear using **Redux**, we can make sure that it is mutated the way we intend it since components cannot update the state directly. For developing the MVP 4.1, the local state provides enough functionality, and **Redux** or other global state management library is not necessary.

[14] *Redux: Motivation* 2022

1.6 JavaScript bundlers and Webpack

JavaScript 4.1 bundler is a tool that gathers all your JavaScript code and its dependencies (npm packages, CSS 4.1 stylesheets, and other types of files) and outputs a new JavaScript 4.1 file, commonly known as a bundle file. This process is called transpilation, and bundlers generally require instructions or configuration files.
The most commonly used JavaScript 4.1 bundler is Webpack; it is downloaded 21 million times every week[15]. It transpiles JavaScript 4.1 using the Babel library so that the output JavaScript 4.1 bundle is compatible with a particular **ECMAScript** 1.1.4 release. That means even old browsers can open a web application converted this way.

1.6.1 PatternFly library

PatternFly[16] is an open-source design system built by **Red Hat**[17] to provide a consistent user experience. It contains design guidelines, user interface layouts, ready-to-use HTML 4.1, and **React** components including charts and tables. Because of its practical approach and high parameterization, it was considered a suitable library for this project.

HTML to PatternFly comparison

Using HTML 4.1 with a combination of CSS 4.1 to create a stylized button, implies creation of two different files: first in the HTML 4.1 format to describe the structure of a component 4.1 and the second CSS 4.1 one to define its styles. Figure 1.12 is an example of how a user could create an HTML button with CSS styles inside. In comparison, the Button component 4.1 from the **PatternFly** 1.6.1 library already includes the CSS 4.1 stylesheets and, at the same time, provides **React** properties that one can use to parameterize the component 4.1 further. Figure 1.13 shows how the **PatternFly** component button could be used in the functional component.

```
1     <button class="btn style="width: 50px; color: green;">
2      <p>Click me</p>
3     </button>
```

Figure 1.12: Basic HTML button with CSS styles inside

Though it requires additional considerations (like, at least, the usage of **React** library and skills to import NPM 4.1 packages), it certainly can be considered a more straightforward approach when there is no demand to use custom stylization. Usage of component libraries simplifies the development process and helps the graphical part be consistent throughout the application.

[15] Webpack page on npmjs.com
[16] PatternFly website
[17] Red Hat

```
import React from 'react';
import { Button } from '@patternfly/react-core';

const ExampleComponent = () => {
 return (
  <Button>
   Click me
  </Button>
 )
}

export default ExampleComponent
```

Figure 1.13: Basic PatternFly Button component

As for other components, **PatternFly** documentation also has a separate web page for the Button component. Besides that, it contains extensive documentation of how the Button component can be modified with parameters. If the parameters do not fulfill some of the styling requirements, these exceptional components can still be supplemented with custom CSS 4.1.

PatternFly documentation has a separate web page for the Button component. It contains the description for all the components available, and if we need to change the style of one of them, we could do that separately using CSS. [18]

1.7 Party System Fragmentation

Internal Encyclopedia of Political Science says that "A party system is fragmented if it contains more than two parties, none of which comes close to obtaining an absolute majority in the representative assembly. Party system fragmentation thus has two aspects: the number of parties in the system and their relative size." (Bertrand Badie and Morlino 2011, page 1822)

To fall under the umbrella term of **"Fragmented Party System"**, a party system should not be a system with one or two parties that get the most significant representation. Furthermore, the party should be represented in the government in order to be documented and analyzed.

Political science researchers use measures represented by equations such as the **Laakso-Taagapera, Dunleavy, Boucek, or Golosov's** formulas to measure system fragmentation. I covered each equation in detail in the following chapters and will use all of them in my prototype. The most well known example of the two-party system is the United States of America. Due to its age and efficiency, all small parties represented in the past joined the most successful ones and fell under one of two parties: democrats or republicans.

On the other hand, countries like the Czech Republic have more considerable party fragmentation and smaller representation. **I calculated the Effective Number of Parties**, and the average number is 4.87, which points out that the party system is highly fragmented. The fragmentation of a party system plays a big role in the comparative literature on political parties. Computer technology allowed us to create robust empirical generalizations.

Political science separates systems by fragmentation into two groups, **Single-Member plurality(SMP)** systems, and **Proportional(PR) list** systems, by the mean effective number of parties since 1960. The average effective number of parties for the **SMP** systems is 2.1. For the **PR** list systems, it is 3.9.

1.7.1 The effect of fragmentation

It is almost impossible for a small party in the **SMP** system to win seats and be represented. On the other end of the spectrum is the **PR** list system, where even a small party could get a representation.
Furthermore, this is affected by the voter's awareness that their vote for the minor party in the **SMP** system will not bring this party representation. Everyone wants to vote for the "winner," so they are more likely to choose one of the two bigger parties. However, the

[18] *Patternfly button documentation* 2022

heavily fragmented system offers the voter a choice to reward a small party with a vote and be represented. Voters are more inclined to give their vote to the small party and stay with it longer.

The more fragmented the system, the more likely the cabinet would have to group into the coalitions, and the shorter these coalitions live. As a voter, you cannot ”blame” one party for the decision if that party is a part of a bigger coalition. Thus it is hard to distinguish against whom you should vote. Fragmentation causes a vulnerability in the political system that non-democratic political forces could use to their benefit.

We could conclude this with a citation from the **Internal Encyclopedia of Political Science**. ”In terms of democratic qualities, fragmented systems tend to perform poorly on accountability but well on representativeness.”(Bertrand Badie and Morlino 2011, page 1823)

1.7.2 Political parties and their functions

”Political parties are one of the few institutions that the birth of which is inextricably linked with the genesis of liberal democracy.” (Golosov 2001, page 151).

I formulated the list of functions of the political parties based on the **Comparative Politology** textbook (Golosov 2001, page 153).

1. The relationship between ruling and control. The party always acts as a channel for the transmission of information between voters and the elected official that circulates ”top-down” and ”bottom-up”. The party thus expresses the social interests of society.

2. Accumulation of social interests. The party always has different and diverse interests, preferences and requirements. In this case, the party acts as a filter, so the summary of the most important interests is presented in the form of a political program.

3. Setting the party collective goals. It is a mistake to think that a party is only able to pursue goals that have already been expressed and exist as thoughts. The party can set goals for society and inspire the masses to implement programs to achieve those goals.

4. Co-optation 4.1 of the new members of the ruling elite. Recruitment must be understood as the selection of staff both for the party itself and for other organizations that are part of the political system, including the appointment of candidates for governor.

5. The role of the party as a reference group. It allows individuals to target specific behaviours. In many countries, people who follow family traditions feel one way or the other.

6. Structuring of the electoral process. Basic functions in liberal democracy. Without parties, elections become more competitive among individuals rather than political agendas.

1.8 Analysis Methods

1.8.1 Formula Markku Laakso and Rein Taagepera

An analysis of the effective number of parties in a country is a formula developed by **Markku Laakso** and **Rein Taagepera**.
"Rather than take the number of all existing parties, including even the very smallest, one visibly has a need for a number that takes into account their relative size. We will call this number the effective number of parties," (Marrku and Taagepera 1979)

$$N_l t = \frac{1}{\sum_1^x s_i^2} \tag{1.1}$$

Where **x** is the number of parties with at least one vote per seat and **s** the square of each party's proportion of all votes and seats. If the resulting number is less than two, then we are most likely a one-party party system; approximately equal to two - a system of two sides; 2.5 or slightly higher - 2.5 parties. With its function, this formula "weighs" each component and determines its attribute weight, thus eliminating the smallest minor components.

To use this formula, we must become familiar with its three main properties.

1. If the components of the selected batch group are the same, then the number N equals the number of these components.
2. If the components of the selected batch group do not match, the number N will be less than the number of these components.
3. If any of the components of the selected dose group decreases, then the number N increases.

It should be emphasized immediately that the use of this type of formula is associated with a loss of information. The loss of information is offset by the use o f quantitative indicators, which allow us to introduce elements of mathematical analysis into mathematics and thus bring it closer to the ideal of a scientific nature.
It is, also possible to create almost complete statistical databases for individual countries and to work with these data in comparative studies.

1.8.2 Dunleavy and Boucek formula

In 2003, two researchers identified problems with the above formula **N(lt)** and proposed an improved version of the calculation of the effective number of batches.

$$N_b = (\frac{1}{\sum_1^x s_i^2} + \frac{1}{s_i}) * \frac{1}{2} \tag{1.2}$$

In short, this formula corrects inaccuracies in the formulas of **Markku Laakso** and **Rein Taagepera**. The quote "misleading results can be obtained, especially if these data are correlated or regressed with other variables in the quantitative analysis." (Dunleavy and Boucek 2003)

1.8.3 Golosov formula

The new approach, written by **Grigory Golosov**, considers a new principle for constructing a formula for calculating the effective number of parties.

$$N_b = \sum_1^x \frac{bs_i}{bs_i + d} \tag{1.3}$$

Where **b** is an arbitrary factor and **d** is the rate of deviation of the size of the **i** (i-th) component (its difference from the size of the first component). If there is no deviation **d = 0** for all components and the index equals the number of components.
According to his research and calculations, a new formula was written to calculate the **Effective Number of Parties**.

$$N_p = \sum_1^x * \frac{\sum_1^x s^i}{\sum_1^x s_i + (\frac{s_1^2}{s_i}) - s_i} \tag{1.4}$$

$$\sum_1^x s^i \tag{1.5}$$

Represents the total number of votes or "seats" distributed during the voting. **Golosov's** method solves several computational problems that political scientists encounter when using the **Dunleavy-Boucek** formula.

This formula meets all the requirements of the original formula of **Markku Laakso** and **Rein Taagepera**

1. It provides reasonable results when counting several important parts.
2. Records the exact number of participants in groups of several participants.
3. It minimizes the so-called "breakthrough" effect of the **Dunleavy-Boucek** formula.
4. The combination of these features indicates a more efficient amount of data.

For the calculation, let's take the last elections of 2017 in the Czech Republic.

Political party	Percent of votes
ANO	29.64
ODS	11.32
PIRATI	10.79
SPD	10.64
KSČM	7.76
ČSSD	7.27
KDU-ČSL	5.80
TOP 09	5.31
STAN	5.18
SVOBODNI	1.56

$$N_l t = \frac{1}{(0.087 + 0.012 + 0.011 + 0.005 + 0.005 + 0.003 + 0.002 + 0.002 + 0.0002)} = 6.973 \tag{1.6}$$

$$N_b = (6.973 + \frac{1}{0.296}) * \frac{1}{2} = 5.176 \tag{1.7}$$

$$N_p = 5.73 \tag{1.8}$$

We therefore count the effective number of parties in the Czech Republic at the time of 2017. Below is a table of calculations[19] and Figure 1.14 describing all elections held in the Czech Republic to see if the number of parties has increased or decreased.

[19] The Google Sheet document containing the calculations

Year	$(N_l t)$	(N_b)	(N_p)
1990	3.57	2.79	1.92
1992	7.65	5.5	4.2
1996	5.41	4.39	3.93
1998	4.73	3.91	3.9
2002	4.84	4.07	3.74
2006	3.92	3.37	3.33
2010	6.8	5.67	6.09
2013	7.68	6.29	7.09
2017	6.97	5.17	5.03
2021	5.26	4.43	4.40

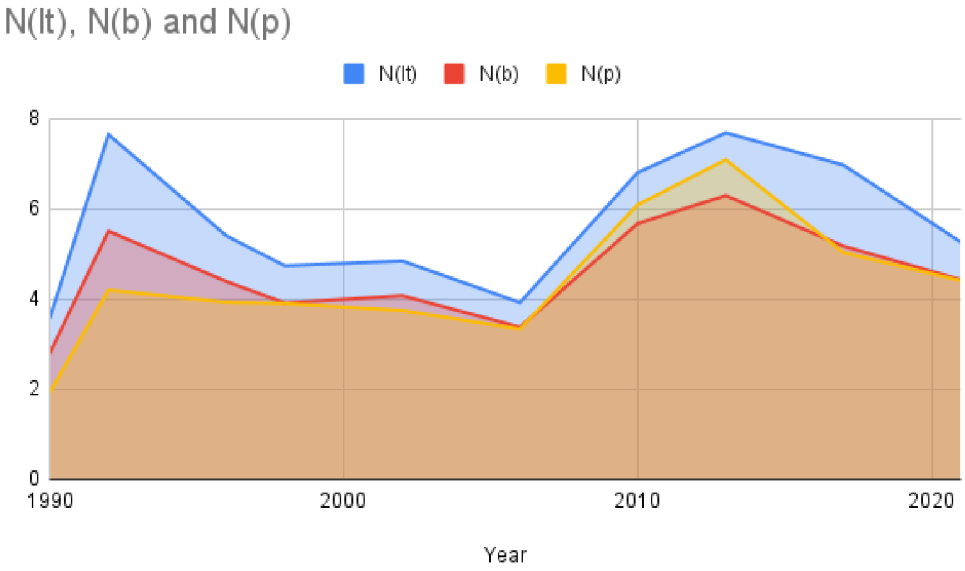

Figure 1.14: Effective Number of Parties in Czech Republic(1990-2021)

Chapter 2

User Experience process

User Interface 4.1 aims to attract people to a site they are interested in; then, once they are there, make their journey from the homepage to purchasing the product or using the service as easy as possible. [1]

2.1 User Research

To understand the user, we need to answer the following questions

1. Who is my user?
2. What are the behavior of my users?
3. What are the needs, pains and requirements of my users?

As part of the research, I have consulted a user group representative **Vladimir Kudryavtsev**[2] from the Department of Political Science of the European University of St. Petersburg[3]. That allowed me to conduct a small User Research and better define the average political scientist or student's needs, pains, and requirements.

2.1.1 User Needs

1. Free to use website with electoral data.
2. Ability to download **CSV** with processed data for further calculations.
3. Political science calculations and formulas.
4. Ability to filter electoral history by date, country, and party name.

[1] Allabarton 2022
[2] Vladimir Kudryavtsev
[3] Department of Political Science of the European University of St. Petersburg

2.1.2 User Pains

1. To do my research, I need to copy data from **Wikipedia** and process it using **Excel** manually.
2. Most of the archives or services cover the US electoral history.
3. Most archives or services are slow and have an old user interface.

2.1.3 User stories

Common User Experience 4.1 practice is to define the user requirements in the form of **User Stories**. They determine the scenario user will go through while using web applications and describe them in several sentences. In the scope of the web application, I am building the user story will be: ”As a student/researcher studying political science, I need a service that could quickly provide me with information on the electoral history for my research”

2.2 User Experience Requirements

According to common practice in the user experience, we need to separate web application requirements into functional and non-functional ones. [4]

2.2.1 The importance of separating requirements

The main goal of separating requirements into the two categories is defining the scope and understanding the requirements web applications should meet to provide functionality and a great user experience.
The customer and a representative of the development team should discuss which features to include in the first place to create a **MVP** 4.1. If the requirements are not defined and constantly changing, the development team has to prolong development time, and the cost of the web application increases.

2.3 Functional Requirements

1. **Business requirements** - create a free access web application that stores electoral data for most countries. This project should be released as open-source so more people can contribute to it in the future.

[4] Puzhevich 2021

2. **User requirements** - users should have access to the graphical user interface that would allow users to interact with it and choose which data to show on the page. A graphical interface should visualize data in a way that would be appropriate. The primary way to visualize it should be a table with a bar chart. The web application should provide a way to download shown data as a **CSV** file.

3. **System requirements** - The web application should work on the most popular browsers such as Chrome, Safari, Firefox, Edge, and Brave.

2.4 Non-Functional Requirements

1. **Usability** - I will keep up with the predefined Red Hat standards and guidelines to improve user interface usability. [5]

2. **Availability**- In the future, I will publish my web application on digitalocean.com or another hosting provider, which will make it accessible 24/7.

3. **Reliability** - **React** 1.2 library allows using the same browser page for hours, so reliability should not be an issue. However, the web application will be tested automatically to confirm that.

4. **Recoverability** - The user will work with immutable data received from the database. The data shown in the bar chart will be stored in the local state and will persist even if the user opens another page and then returns to the table.

5. **Scalability** - As I mentioned before, I plan to host the website on services such as digitalocean.com. Hosting provider for small web applications with the opportunity to scale it up.

6. **Performance** - The web application based on **React** 1.2 will work on the **Single Page Application (SPA)** 1.2.2 principle that was covered in the Theory section 1.2.2. **SPA** 1.2.2 removes almost all of the loading time from the web application, so the loading times become seamless. Further optimization could be done to reduce the JavaScript 4.1 bundle sizes that the user receives when he opens the page for the first time.

7. **Supportability** - The web application will have its repository on **GitHub** 4.1, so other people can participate in further development down the road. In order to fix the bugs and issues with the web application, they will be reported and documented. After they are fixed new version of the web application will be released and published.

8. **Security** - The current version of the web application will not contain personal user data. The rest of the information is publicly accessible and could be used by anyone.

9. **Capacity** - The web application capacity is restricted by the hosting tariff digitalocean.com provides, and could be scaled up.

[5] *Patternfly UX writing* 2022

2.5 Figma mockup

To develop the **User Interface** 4.1, you need to understand it and visualize it based on the User Experience 4.1 research. One of the best solutions that help **User Experience** 4.1 Designers create mockups is a program called **Figma**. This software provides easy ways to prototype the interface and run scenarios based on the "user stories". You can use component libraries like **Ant Design**, **Material UI**, or **PatternFl**y to create and edit mockups. The Figures 4.1, 4.2, 4.3 and 4.4 visualize basic scenario how user would interact with the **User Interface** 4.1.

Chapter 3

Implementation

3.1 Prototype development

Throughout the development process I have tried three technology stacks:

1. Gatsby.JS with GraphQL.
2. React 1.2 with D3 library.
3. React 1.2 with PatternFly 1.6.1 library.

Gatsby.JS has the same issue as any "closed" development ecosystem. It is tough to implement anything unusual, and there is no data visualization plugin in **Gatsby.JS**. It is better suited for the development of small-sized projects and web pages like portfolios or blogs.
D3 Library is a vast and complex library that requires a deep understanding of how SVG images are generated in the browser. It would be a better solution for the final product but is too complex to create prototypes.
In the end, I have chosen **React** 1.2 and **PatternFly** 1.6.1 because they let me create a working prototype faster. Moreover, the **PatternFly** 1.6.1 template enabled me to use **TypeScript** 1.1.5 , which opened the possibility of typization and allowed me to devise better data schemes.

3.2 Create a web app using the PatternFly template

PatternFly 1.6.1 library provides a **TypeScript** 1.1.5 template for creating User Interface 4.1. It contains several essential libraries you require for the development using **React** and contains **Webpack** 1.6. [1]

[1] *Develop with PatternFly* 2022

Using the “Getting started” instruction, I have installed, created a project, and connected it to the repository on **GitHub**[2] 4.1.

3.2.1 Visual diagram of the components tree

Figure 3.2 shows how **React Router** wraps the App component and helps the Single Page Application 1.2 sell the impression that it has a lot of “pages.” It gathers the major components into the App Layout component and imports them into the App component. Figure 3.1 describes the component structure of the application. To add a ”page,” you need to define a new route in the **routes.tsx** file. A routes array contains objects describing the information that **Router** requires to direct the user correctly. You need to import the component and add it to the array’s ”component” property.

```
import * as React from 'react';
import '@patternfly/react-core/dist/styles/base.css';
import { BrowserRouter as Router } from 'react-router-dom';
import { AppLayout } from '@app/AppLayout/AppLayout';
import { AppRoutes } from '@app/routes';
import '@app/app.css';

const App: React.FunctionComponent = () => (
  <Router>
    <AppLayout>
      <AppRoutes />
    </AppLayout>
  </Router>
);

export default App;
```

Figure 3.1: App Routes help us to define the routes in our application. It would instruct the **React Router** which component should be rendered when the user opens the Dashboard, Settings, or other ”Page.”

[2] GitHub

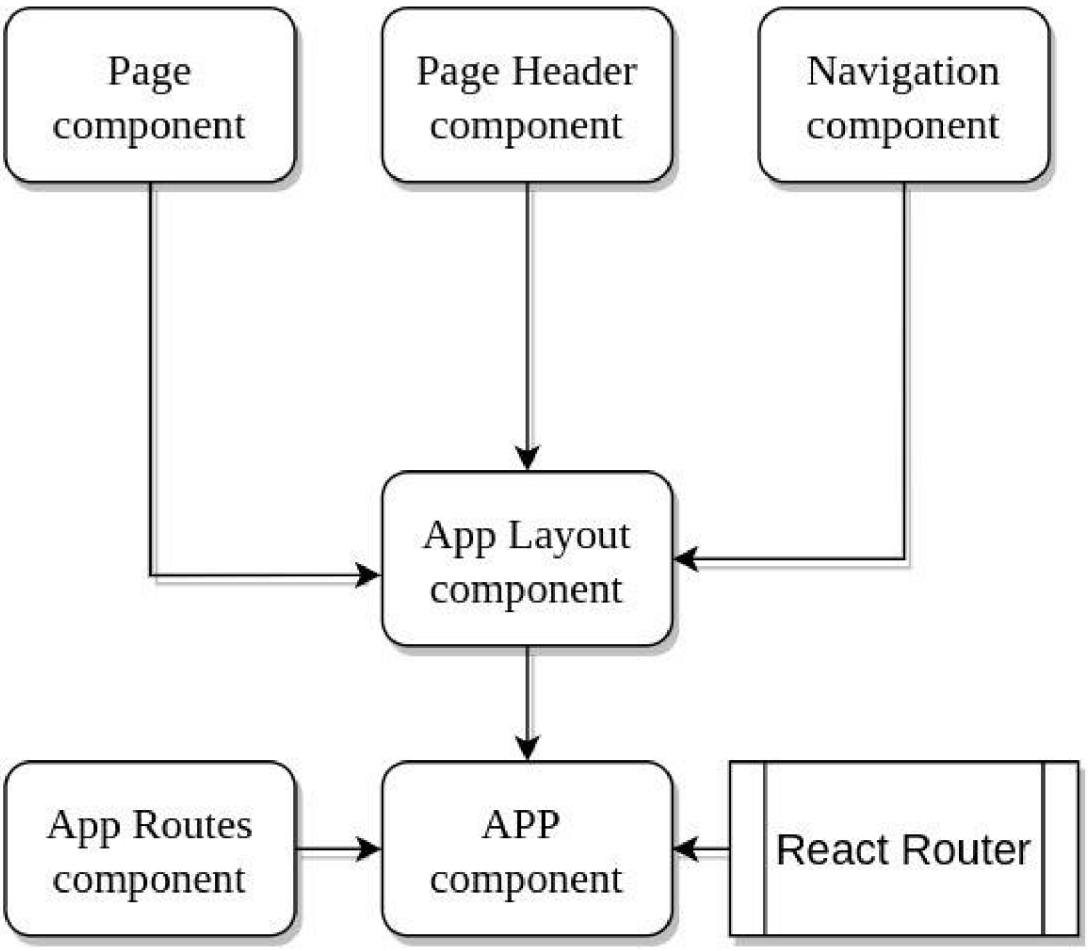

Figure 3.2: PatternFly template component tree of the application

3.2.2 How data is stored

Because the first stage of development is to create an **MVP** 4.1, I have decided to use JavaScript 4.1 object structure to store the data and import it directly to process and pass it to their children's components. I used the **TypeScript** 1.1.5 type interface to define data types in objects and made it less error-prone and easy to understand. Figure 3.3 describes the interface I created

```
export interface CountryTable {
  name: string;
  electionRecords: number | null;
  numberOfParties: number | null;
  countryDetails?: {
    detail1?: string;
    detail2?: string;
    detail3?: string;
    detailFormat: 0 | 1 | 2 | 3;
  };
  seats: number;
  electionYears: any[];
  electionDetails: any[];
}
```

Figure 3.3: Country table data structure declared using TypeScript interface.

3.3 Component Structure

3.3.1 Dashboard Component

The Dashboard component 4.1 represents the Default welcome page. It contains instructions on using the website and describes what it does.

3.3.2 Country Page component tree

The country page component 4.1 is our so-called **"container"** component 4.1 that combines all parts. To get easier access to the Bar Chart state and data from the Country Page Table, I had to "lift" the state to the Country Page component 4.1. That allows easy access to any children component 4.1 of the Country Page. The Figure 3.4 visualizes the Country Page component tree. Main functions of the Country Page component.

1. It contains the local state for the Bar Chart and stores the data about parties after the user adds the data to the Bar Chart.
2. Combines the data from the two sources to prepare it to be downloaded as **CSV**.
3. Serves as a container for all of its children's components.

3.3.3 Bar Chart component

This component is responsible for rendering the data passed from the Country Page. The useEffect **React** hook is responsible for calling a groupData function that sorts the data into three different arrays and updates the averagePartyData state in the Country Page component. After that, it creates one bar for the chart for each index in these arrays.

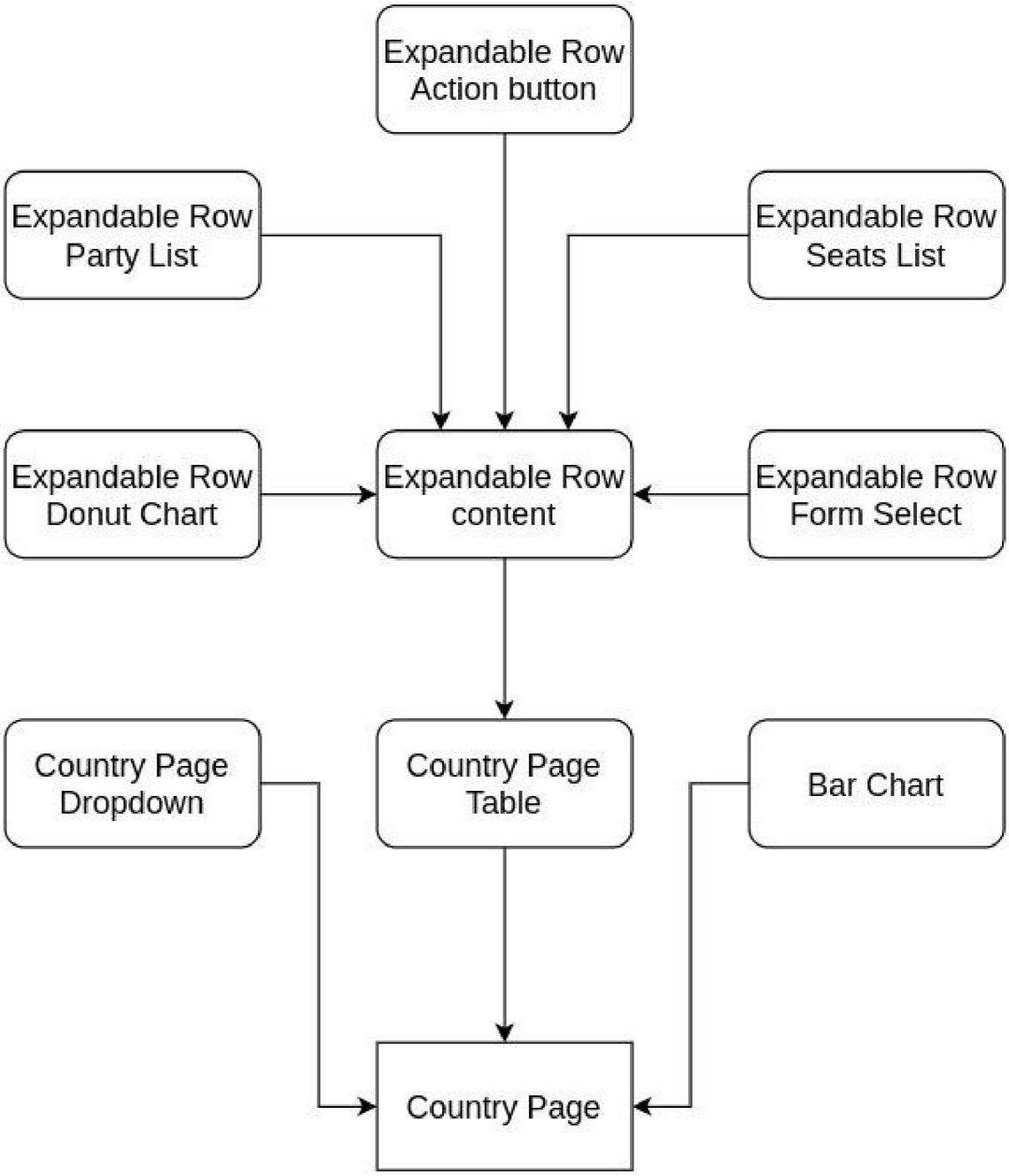

Figure 3.4: Country Page component tree

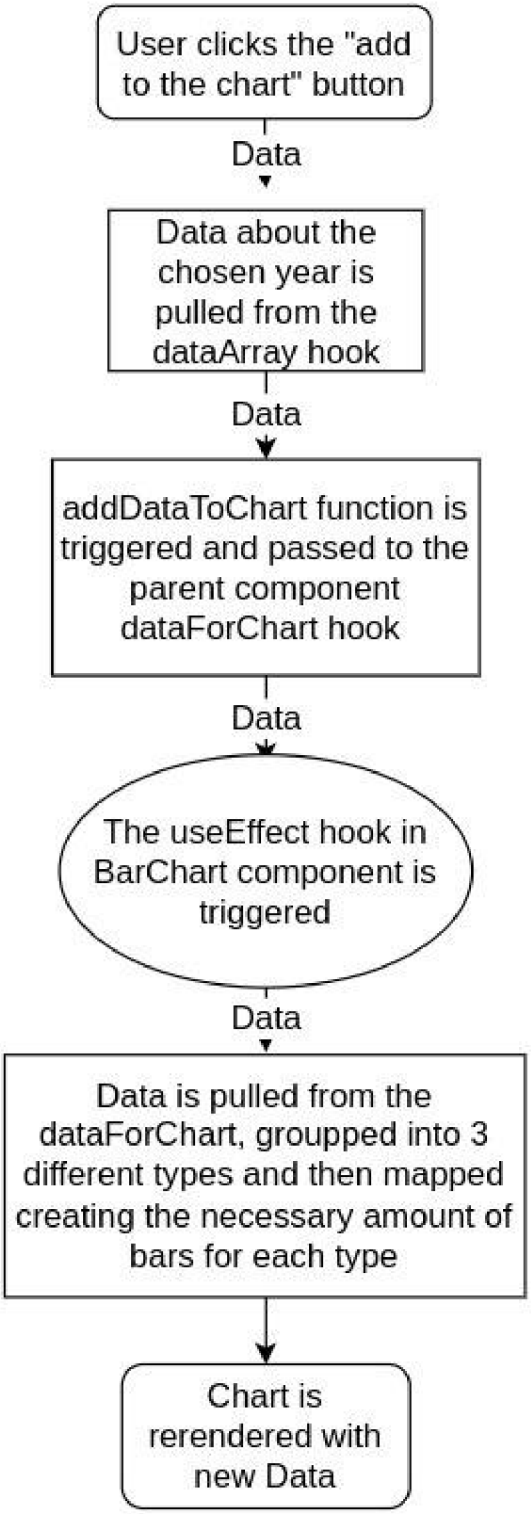

Figure 3.5: This diagram visualizes how the data is passed to the Bar Chart component

UseEffect **React** hook allows us to "subscribe" functions to the change in the Country Page state. All of the operations mentioned above are firing automatically, and a user could add the data to the Chart and Clear it without any issues. The Figure 3.5 visualizes the data flow of the Bar Chart component.

3.3.4 Country Page Dropdown component

This dropdown component collects all the actions user could do with the Bar Chart. Each dropdown item is a component imported from the **PatternFly** 1.2 library, and they are described in an array. There are two main functions that users could apply to the Bar Chart.

1. Clear the graph by clicking on the "Clear the graph" button in the dropdown. That would trigger cleanBarChart function and will remove all the data from the dataForChart.
2. Download the visualized data as a **CSV** file.

3.3.5 Processing data for CSV

To prepare the data to be downloaded as **CSV**, I had to write a custom function that combines the visualized data with the information about the party votes and how many seats each party got.

1. I declared the collectingData array using the useState React hook. This array is updated when the user adds data to the chart.
2. I wrote an useEffect hook to store the data and subscribed it to the dataForChart. That made the collectingData update and store the previous and new data.
3. I declared the readyData as an empty array, assigned the collectingData to the flattenData, and used the .flat(1) method to reduce the nesting.
4. For each nested array in the flattenData, push the new object into the readyData and assign data to the object parameters.
5. I create a custom filter function that receives the array, name, and year as function parameters and returns the array based on the name and year parameters.
6. After all these steps, our data is prepared to be combined into the new objects, pushed into the data **CSV** array, and passed to the onClick effect on the **"Export graph as CSV"** button.

3.3.6 Country Page Table

It is a component responsible for the table with data. To fill in the table rows, we import the data and then process it, creating a new row and filling in the select menu for each country in the imported file. The Figure 3.6 show the example of it. The Country Page Table component received the setDataForChart method from the Country Page component and used it to power the "addDataToChart" function. Some functionality, such as

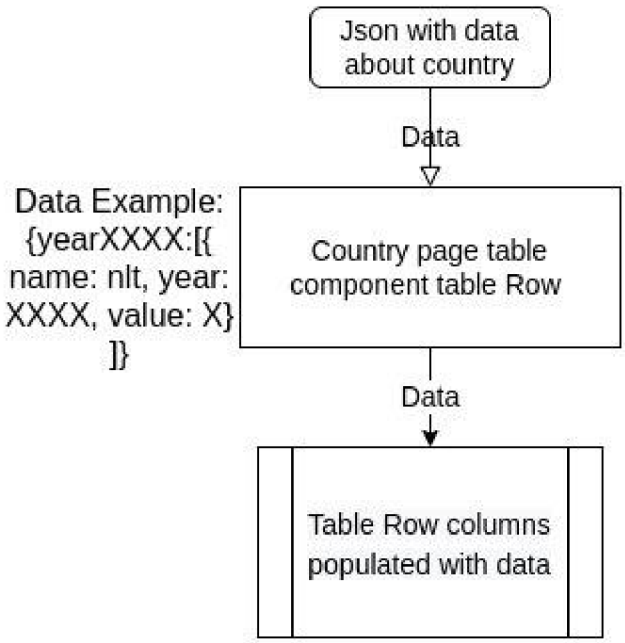

Figure 3.6: Example of how the Table's row are populated with data

pagination, filtration and search were not necessary to develop a prototype, and is missing. The missing parts should be added after an actual working database is ready to store the data.

Table Row Expanded

When the user clicks on an arrow symbol on the left side of the row, it will expand, showing the following components.

1. Select a form that allows users to choose a year from the dropdown.
2. Donut chart to visualize the year chosen in the form.
3. The Column that shows the percent of votes each party received in the chosen year
4. Amount of seats in the parliament party received in the chosen year
5. Action button that allows users to add the data to the bar chart

Form Select component This component is responsible for two things:

1. Allows the user to choose the year he wants to visualize.
2. Filters the data depending on the chosen year and updates the electionDataDetails in the Country Page Table component

Donut Chart component This component receives the electionDataDetails from the Country Page Table component and renders it.

Party List component This component receives the electionDataDetails from the Country Page Table component and renders it.

Seats List component This component receives the electionDataDetails from the Country Page Table component and renders it.

3.4 The second prototype in Figma

During the development process, I had to change the first design I came up with and update it accordingly. After I finished developing the MVP 4.1, I updated the mockup to better understand what the page should look like. Figures 4.1, 4.6, 4.7 and 4.8 shows the updated mockup for the User Interface 4.1.

Chapter 4

Results

This study aimed to develop a user-friendly platform that helps anyone acquire, create, and share accurate scientifically verifiable knowledge about electoral history. Nowadays, especially in the light of current events, it is hard to overestimate the impact **political literacy** has on a given political situation worldwide. For example, it is common for oppressive political regimes to pray on an illiterate electorate and abuse the power they usurp.
During the project we identified user user requirements, design data storage and user interface that required to coordinate experts in user experience and political studies. To evaluate the basic idea and validate the design, we implemented a prototype that was later transformed into a MVP 4.1. The product has only few core features but is functional, usable, and ready for testing and further development. It has a visual layer that in a clear manner presents several important indicators, such as number of parties, their results in election, and measure of party system fragmentation with simple User Interface 4.1 which can be used by any person without special knowledge. It also provides the more advanced users with the unified access to raw historical data from different countries for further analysis.
The platform accessible to anyone as a web page. The source code is shared on **GitHub**. That way, new developers could propose new features and improvements or branch off their version. Moreover, all of the data shown is publicly available on Wikipedia or other resources. By making the data freely available in a simple, clear and user-friendly manner, the project has potential to increase involvement into politics and the level of political literacy, thus bringing us a step closer to a more stable society.
After the **MVP** 4.1 development is finished, the real work has just started. The next goal is to validate the basic assumptions, identify the right direction for the development, and finally bring the service closer to the first major release. I do not plan to abandon this project, especially now, when so many people have put their time and effort into this concept and the future development team is already being gathered. In the next section we will discuss existing implementation's problems together with further development possibilities.

4.1 Next steps

To make the platform more scalable and robust would require to change the way the data is stored and cover application with tests. On the other side, with existing MVP we already have first user feedback about the product and identified required features for the next stage of the development process. One important step is to continue collect the feedback, both qualitative and quantitative. Other logical next steps in the platform development path could include:

1. To design and implement PostgreSQL database as was advised by my colleague from **Red Hat - Pavel Tisnovsky** [1], who is an experienced database engineer. **PostgreSQL** is a very scalable database system, that also has flexible text search and support for **JSON** 4.1.

2. To cover the application with automated tests, in particular to set up tests that should run on each repository update. It helps to ensure means that any bug or inconsistency would be caught before it goes into production.

3. To make the **Country Page table** more user-friendly. That could be achieved by implementing the following features:

    - Filters based on the country, party name and year.
    - Pagination.
    - Make the user choices persist between pages.
    - Allow users to mix data from different countries.
    - Export as PDF.
    - User Authorization.

4. To improve user interactions with the **Bar Chart** by adding automatic chart size scaling, so that the size of the shown bars decreases when the user adds more bars to the screen, and to allow users to choose bar color schemes. Another update would be an automatic reading of year minimum and maximum from the user data.

5. To update **Single Country Page** with more data about each country in our database. That would create a single-entry point for users to get all relevant information on the selected country.

6. To set up an **API** and sell access to it. An API 4.1 should provide third-party organisations and private people the direct access to the data from our server and process it on their side. In the future it may become the independent source of funding for the project and pay for the server. A so-called “embed link” would allow users to create data visualization and add it to their article, website, or science paper using an HTML 4.1 keyframe.

[1] Pavel Tisnovsky

7. To add an **Articles page** in order to make the website closer to the political science students. That would allow students to publish their articles on the website.

8. To add support for **other languages**. The internalization is very important for the overall long-term goals of the project. First, the multi-language support needs to be implemented using a relevant library, like **Format.js** [2]. Next, the platform should start to gather volunteers willing to work on the user interface and data translation.

9. To migrate charts to the **D3 data visualization library**. JavaScript library called D3 [3] seem to be the best solution for more complex data visualisations. It has a much richer functionality that would allow to implement many new features that are out of the scope of the current project. On the other hand, it has a steeper learning curve and thus was not an appropriate choice for prototype and MVP phases.

[2] Format.Js
[3] D3

Acronyms

MPA Multiple-page application. 15, 16

MVP Minimal Viable Product. 8, 9, 11, 12, 22, 24, 34, 39, 45, 46

NPM Node Package Manager. 14, 18, 25

SPA Single-page application. 15–17

Glossary

AJAX Asynchronous JavaScript And XML. 16

API stands for application programming interface, which is a set of definitions and protocols for building and integrating application software "What is an API?" 2017. 19

Co-optation The process of adding members to an elite group at the discretion of members of the body, usually to manage opposition and so maintain the stability of the group "Co-option" 2022. 28

component independent and reusable parts of code. They serve the same purpose as JavaScript functions and return HTML. 18–20, 22–25, 40

CSS CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. 15, 16, 25–27

Document Object Model The DOM represents the web page and gives a way to manipulate it *Introduction to the DOM* 2022. 16, 19

ECMAScript editions ECMAScript editions - or versions are published by TC39 and have been abbreviated to ES1, ES2, ES3, ES5 and ES6. After 2016 the named by year ES2016, ES2017, etc. 14

GitHub a Git repository hosting service that provides a graphical web interface. It is the world's largest community of programmers. 15, 35

HTML The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. 15, 16, 19, 25, 47

HTTP stands for Hyper Text Transfer Protocol. Communication between client computers and web servers is done by sending HTTP requests and receiving HTTP Responses *What is HTTP* 2022. 16

JavaScript JavaScript is a computer programming language. It is lightweight and most commonly used as a part of web pages. 14–16, 18, 19, 25, 35, 39, 48

JSON Stands for JavaScript Object Notation it is a text format for storing and transporting data. 16

PostgreSQL PostgreSQL is an advanced, enterprise-class, and open-source relational database system "What is PostgreSQL?" 2022. 9, 12

User Experience User Experience is the relationship between a product and the person using it. UX design focuses on building products that one can utilize to achieve their goals efficiently. 34, 36

User Interface User Interface is anything a user can interact with to use a digital product or service. 8, 11, 24, 33, 36, 37, 45, 46

Bibliography

Allabarton, Rosie (2022). "What is UX design". In: *Careerfoundry.com/*. URL: https://careerfoundry.com/en/blog/ux-design/the-ux-design-process-an-actionable-guide-to-your-first-job-in-ux/#what-is-ux-design.

Bertrand Badie, Dirk Berg-Schlosser and Leonardo Morlino (2011). *International Encyclopedia of Political Science*. SAGE Publications, Inc.

Brewster, Cordenne (2021). "6 Examples of Where and When You Can Use JavaScript". In: *Trio.dev*. URL: https://trio.dev/blog/examples-javascript.

"Co-option" (2022). In: *Wikipedia.org*. URL: https://en.wikipedia.org/wiki/Co-option.

Components and Props (2022). URL: https://reactjs.org/docs/components-and-props.html.

Context (2022). URL: https://reactjs.org/docs/context.html.

Develop with PatternFly (2022). URL: https://www.patternfly.org/v4/get-started/develop/#start-with-the-react-seed.

Doods, Kent C. (2021). "How to use React Context effectively". In: *Kentcdodds.com*. URL: https://kentcdodds.com/blog/how-to-use-react-context-effectively.

Dunleavy, Patrick and Francoise Boucek (2003). *Constructing the Number of Parties*. DOI: https://doi.org/10.1177%2F1354068803009003002. URL: https://journals.sagepub.com/doi/10.1177/1354068803009003002.

ECMAScript Language Specification (2022). URL: https://tc39.es/ecma262/.

Glossary of React Terms (2022). URL: https://reactjs.org/docs/glossary.html#single-page-application.

Golosov, Grigory (2001). *Comparative Politology*. European University of St. Petersburg. ISBN: ISBN 5-94380-010-7.

Introduction to Electron (2021). URL: https://www.electronjs.org/docs/latest.

Introduction to the DOM (2022). URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction.

Keizer, Gregg (2020). "Google's Chromium browser explained". In: *Computerworld.com*. URL: https://www.computerworld.com/article/3261009/googles-chromium-browser-explained.html.

Marrku, Laakso and Rein Taagepera (1979). *EFFECTIVE NUMBER OF PARTIES A Measure with Application to West Europe*. DOI: https://doi.org/10.1177%2F001041407901200101. URL: https://journals.sagepub.com/doi/10.1177/001041407901200101.

Node.js documentation describing browser engines (2021). URL: https://nodejs.dev/learn/the-v8-javascript-engine.

Patternfly button documentation (2022). URL: https://www.patternfly.org/v4/components/button/.
Patternfly UX writing (2022). URL: https://www.patternfly.org/v4/ux-writing/about/.
Puzhevich, Victoria (2021). "Functional vs Non-Functional Requirements: The Definitive Guide". In: *Scand.com*. URL: https://scand.com/company/blog/functional-vs-non-functional-requirements/.
Redux: Motivation (2022). URL: https://redux.js.org/understanding/thinking-in-redux/motivation.
Single-Page Application vs Multi-Page Application: Pros, Cons, and Which is Better? (2020). URL: shorturl.at/goDI5.
Tate, Darkebe (2015). "What is TypeScript? Pros and Cons". In: *Medium.com*. URL: https://medium.com/@BuildMySite1/what-is-typescript-pros-and-cons-8dc5cdc3e78d.
The Good and the Bad of TypeScript (n.d.). URL: https://www.altexsoft.com/blog/typescript-pros-and-cons/.
v8 Engine documentation (2021). URL: https://v8.dev/docs.
"What is an API?" (2017). In: URL: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces.
What is HTTP (2022). URL: https://www.w3schools.com/whatis/whatis_http.asp.
"What is PostgreSQL?" (2022). In: *Postgresqltutorial.com*. URL: https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/.

4.2 Materials

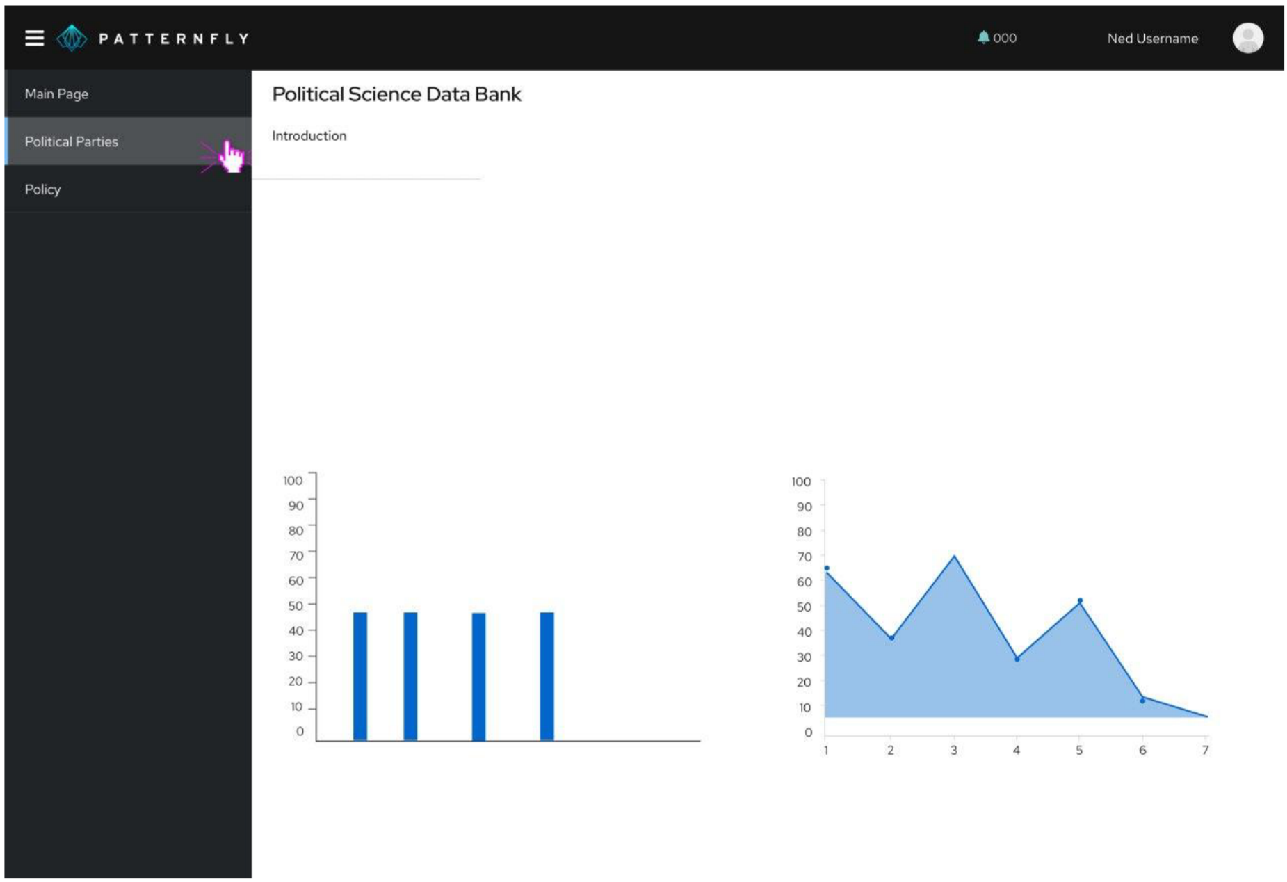

Figure 4.1: User Experience mockup - Dashboard

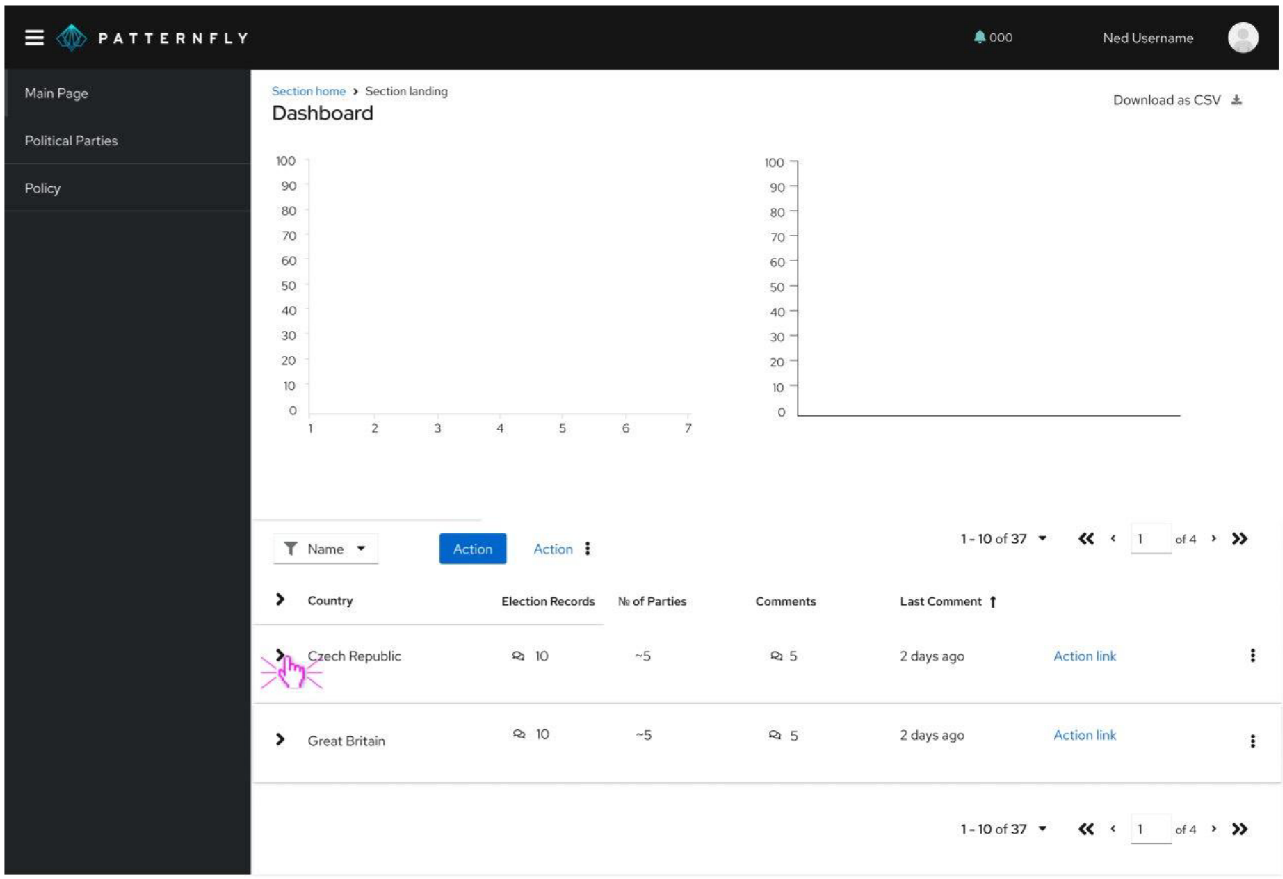

Figure 4.2: User Experience mockup - Country page

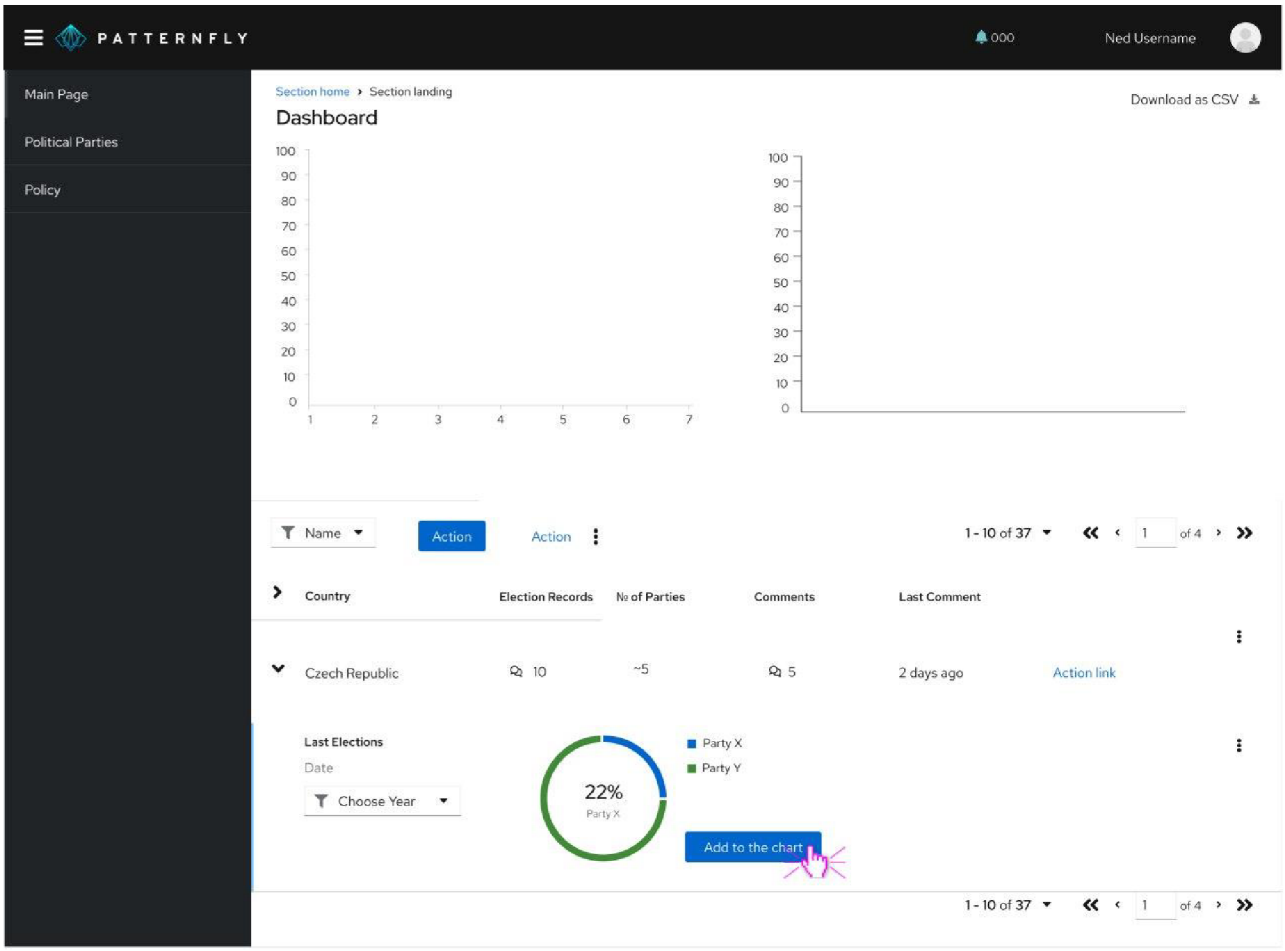

Figure 4.3: User Experience mockup - Country page table row expanded

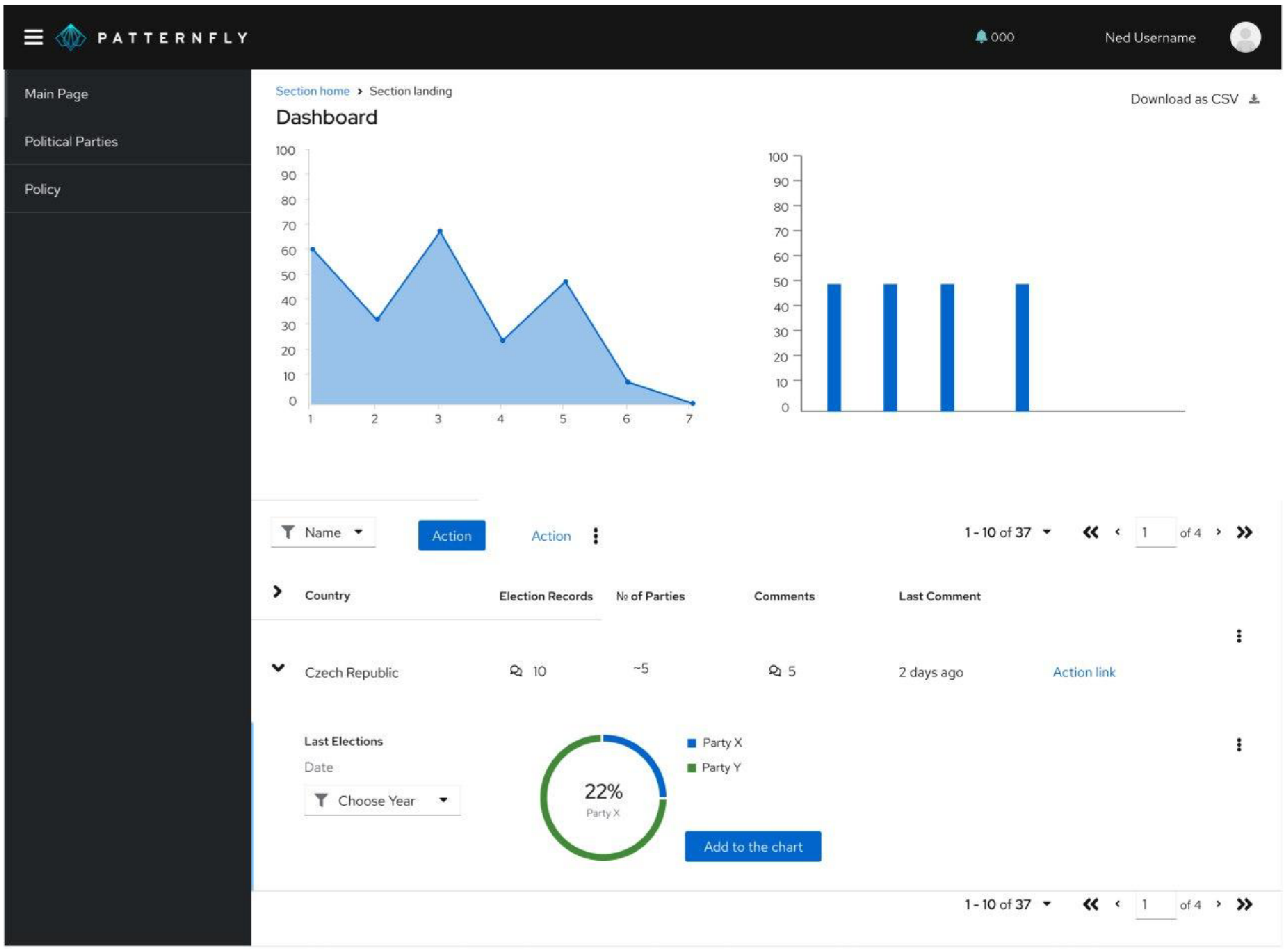

Figure 4.4: User Experience mockup - Country page data visualized

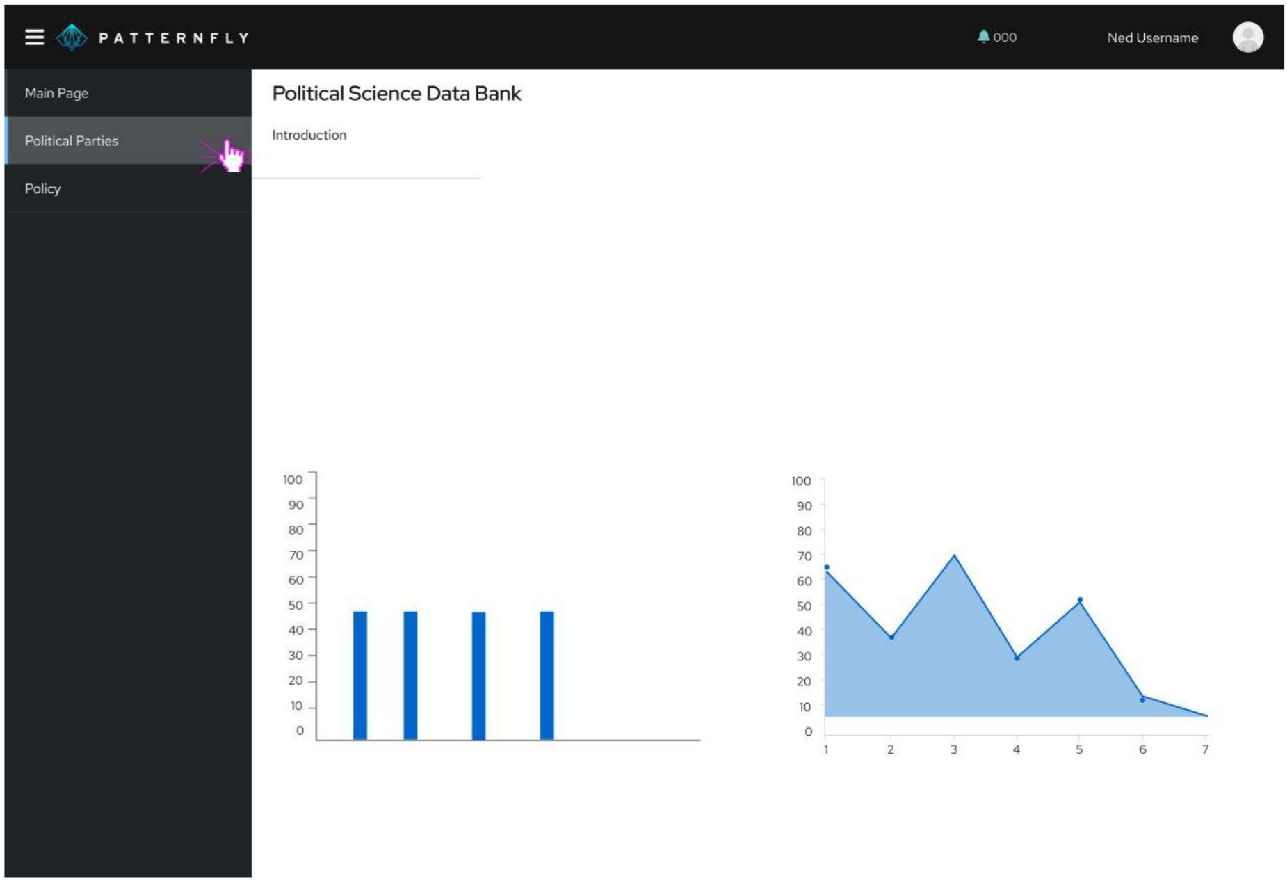

Figure 4.5: User Experience mockup №2 - Dashboard

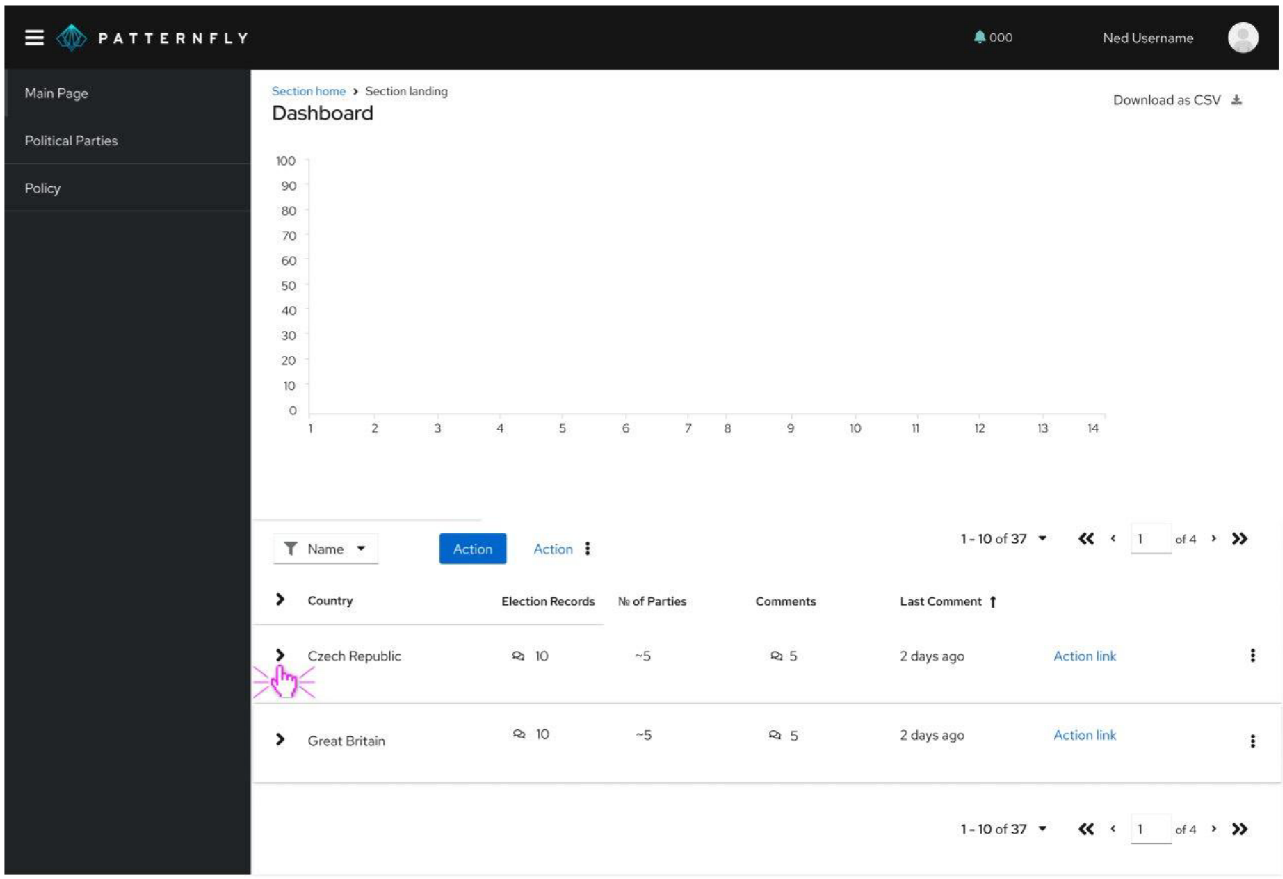

Figure 4.6: User Experience mockup №2 - Country page

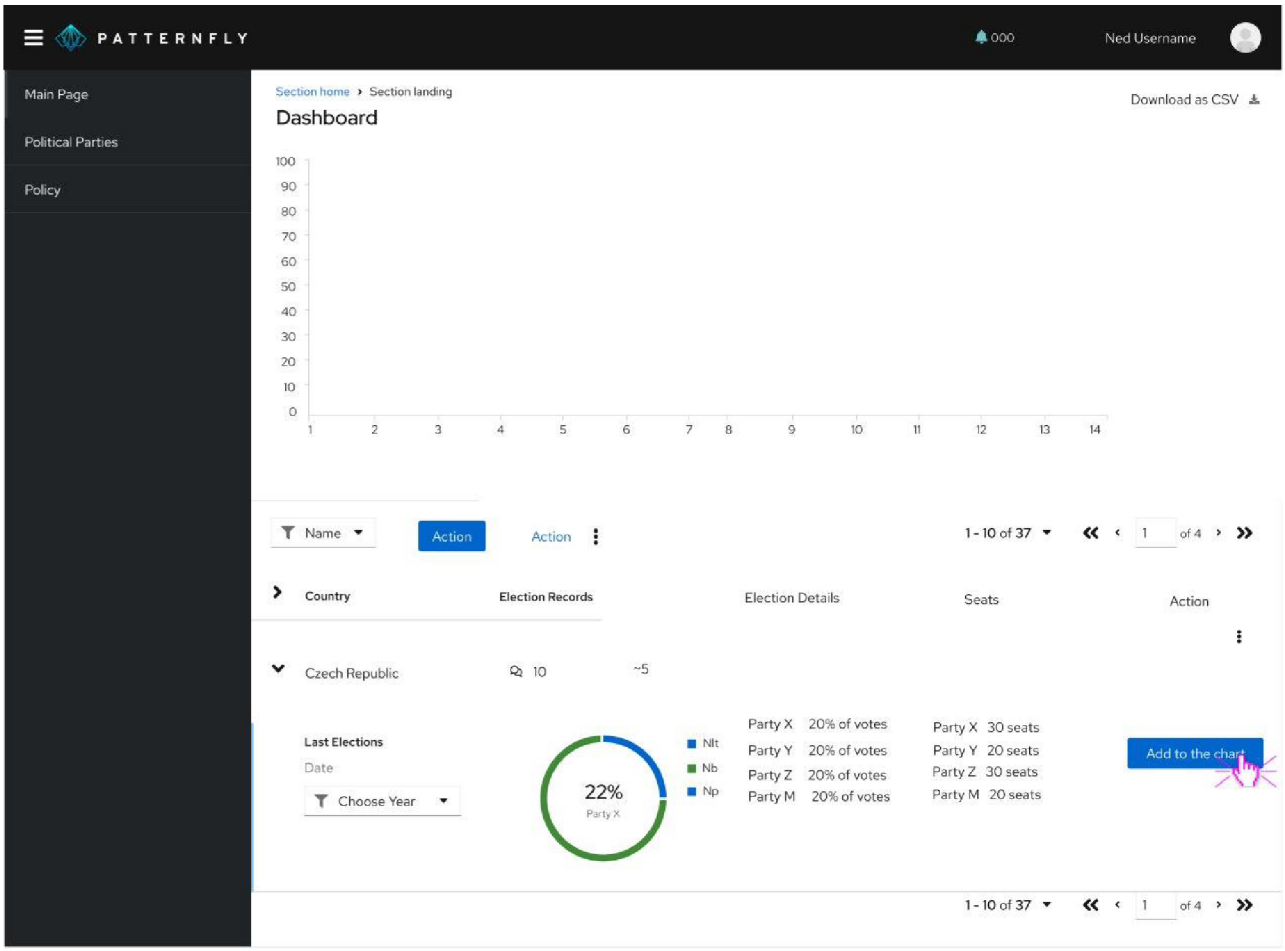

Figure 4.7: User Experience mockup №2 - Country page row expanded

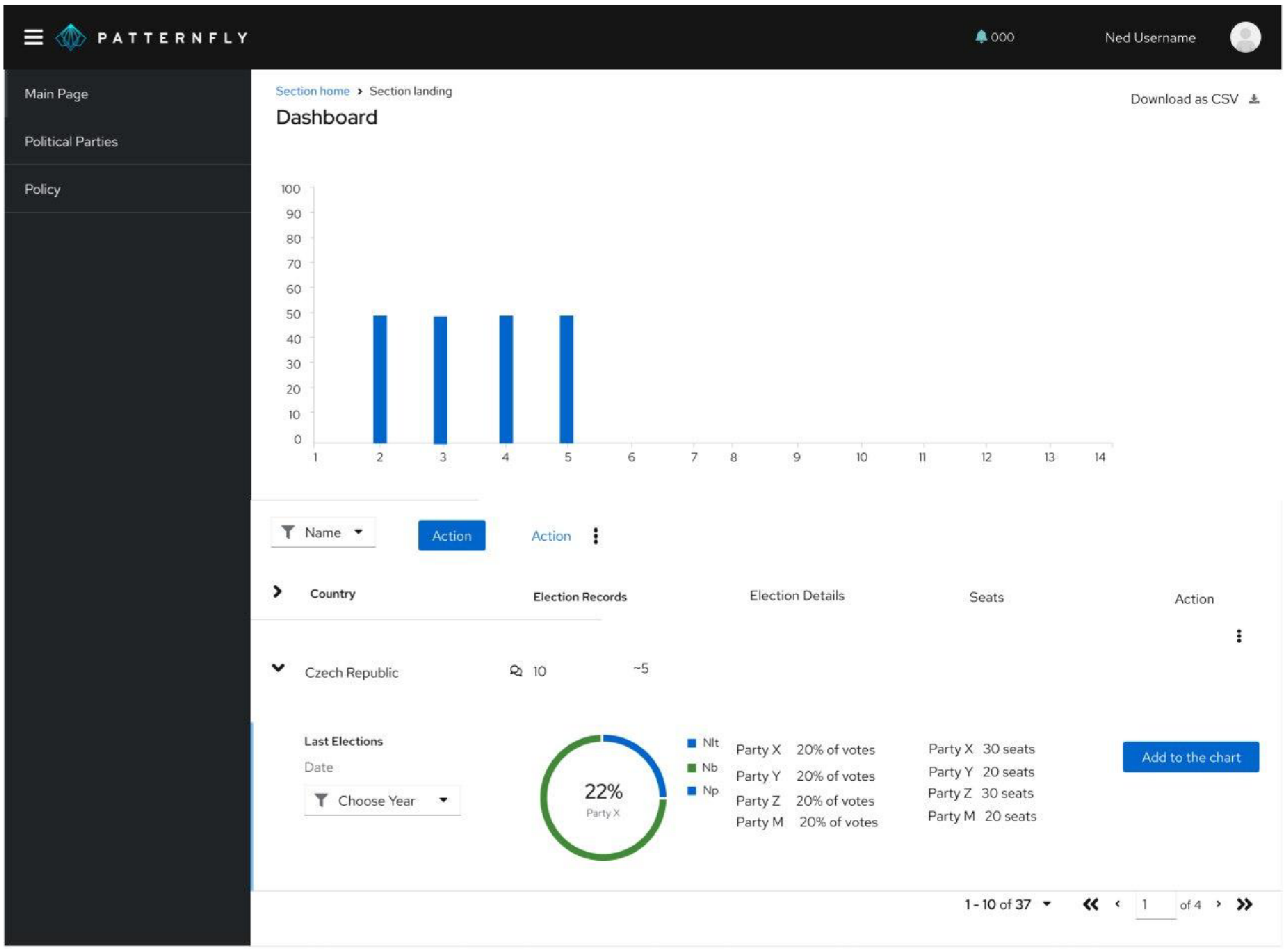

Figure 4.8: User Experience mockup №2 - Country page data visualized