UNIVERSITETI “ISA BOLETINI” MITROVICË

FAKULTETI I INXHINIERISË MEKANIKE DHE KOMPJUTERIKE

DEPARTAMENTI: INFORMATIKË INXHINIERIKE

PUNIM DIPLOME

Mentori: Kandidati:

Prof. Berat Ujkani Diellza Pllana

Mitrovicë, Mars, 2023

UNIVERSITETI “ISA BOLETINI” MITROVICË

FAKULTETI I INXHINIERISË MEKANIKE DHE KOMPJUTERIKE

DEPARTAMENTI: INFORMATIKË INXHINIERIKE

PUNIM DIPLOME

REACT HOOKS

Mentori: Kandidati:

Prof. Berat Ujkani Diellza Pllana

Mitrovicë, Mars, 2023

DEKLARATA E ORIGJINALITETIT / AUTORËSISË

Deklaroj se ky punim është origjinal i imi dhe e tërë puna e bërë në këtë punim diplome është produkt i punës sime individuale dhe hulumtimit tim akademik. Ky punim diplome është përgaditur vetëm për Universitetin “Isa Boletini”, dhe nuk është përdorur në asnjë universitet apo institucion tjetër e as që është marrë si punim i ndonjë personi apo institucioni tjetër. Informacionet e marra janë të gjitha të referuara dhe cituara në burimin që kam marrë.

Po ashtu deklaroj se të gjitha burimet e përdorura janë të miratuara, pas shqyrtimit të kujdesshëm nga ana e mbikëqyrësit tim, Prof. Berat Ujkani. Po ashtu njoftoj që kam respektuar rregullat themelore akademike dhe shkencore të Universitetit “Isa Boletini” për punimin e temës së diplomës.

Mitrovicë 2023,
Diellza Pllana

Përmbajtja

1. Hyrje

Me zhvillimet e shpejta në teknologji dhe numrin gjithnjë në rritje të aplikacioneve në internet, është bërë një domosdoshmëri krijimi i aplikacioneve të shpejta për t'iu përshtatur tregut aktual. Kompleksiteti dhe popullariteti në rritje i aplikacioneve në ueb dhe lëvizja e aplikacioneve të ndërmarrjeve nga arkitektura e bazuar në desktop në arkitekturën e bazuar në ueb kanë krijuar sfida të sofistikuara, por dhe nevojën për arkitekturë moderne.

Një mënyrë për të menaxhuar këtë kompleksitet është të dizajnoni dhe zhvilloni aplikacionin duke përdorur ReactJS dhe arkitekturën e një faqeje (Single Page Application SPA). Krijimi i një aplikacioni të thjeshtë me React nuk konsiderohet kompleks dhe mund të realizohet pa ndihmën e librarive shtesë. Por për zhvillimin e aplikacioneve më komplekse, mijëra librari u publikuan duke ofruar qasje të ndryshme rreth menaxhimit të ‘state’ dhe ‘lifecycle’ të front-end aplikacioneve duke e bërë të vështirë të zgjedhim qasjen e duhur për një projekt.

Deri në versionin 16.8 të React ‘state’ mund të menaxhohej vetëm në komponentet klasa, hooks u publikuan dhe u mundësua që kjo të arrihej dhe me komponentet funksione.Në këtë punim do të ofrohet një analizë e detajizuar rreth React Hooks, llojeve, benefiteve dhe problemeve që hooks zgjidhin, poashtu do të demostrohet përdorimi praktik i React Hooks në aplikacionin ‘Rate Teacher’, aplikacion të cilin e kemi zhvilluar mbi idenë që të ofrojmë tek studentët një platformë ku do të munden të japin komentet dhe kritikat e tyre mbi stafin akademik në institucionet publike dhe private.

2. Hyrje në React

2.1 Çka eshte ReactJS?

React është një librari e JavaScript për ndërtimin e ndërfaqeve të përdoruesit. Mirëmbahet nga Facebook dhe një komunitet zhvilluesish dhe kompanish individuale. React i lejon zhvilluesit të ndërtojnë komponentë UI të ripërdorshme dhe të menaxhojnë gjendjen e aplikacioneve të tyre në një mënyrë efikase [1]. React përdor një DOM virtual për të përmirësuar performancën duke minimizuar numrin e ndryshimeve të bëra në DOM-in aktual. Mund të përdoret në kombinim me librari dhe frameworks të tjera, si Redux, për të menaxhuar gjendjen e aplikacioneve më të mëdha. Megjithëse React është një librari dhe jo një gjuhë, ajo përdoret gjerësisht në zhvillimin e uebit, dhe tani është një nga bibliotekat frontend më të përdorura për zhvillimin e uebit.

React u shfaq për herë të parë nga Facebook në vitin 2013. U zhvillua nga Jordan Walke, një inxhinier softuerësh në Facebook, i cili u frymëzua nga problemet që hasi gjatë ndërtimit të 'news feed' në platformën e Facebook [2]. React u krijua për të adresuar çështjet e performancës së dobët dhe mungesës së mirëmbajtjes që lindën nga ndërtimi i aplikacioneve të mëdha, komplekse në ueb duke përdorur teknika tradicionale.

React fitoi shpejt popullaritet midis zhvilluesve dhe që atëherë është bërë një nga bibliotekat më të përdorura JavaScript për ndërtimin e ndërfaqeve të përdoruesve. Në 2015, React Native u shfaq, i cili lejon zhvilluesit të përdorin React për të ndërtuar aplikacione celulare për iOS dhe Android.

Facebook ka vazhduar të zhvillojë dhe mirëmbajë React, dhe libraria ka pësuar shumë ndryshime dhe përditësime që nga publikimi i saj fillestar. Popullariteti i tij është rritur dhe përdoret gjerësisht nga shumë kompani, duke përfshirë Airbnb, Netflix dhe Uber [3].

Popullariteti i React sot ka tejkaluar atë të të gjitha frameworks të tjera të zhvillimit të front-end. Ja pse:

- Krijimi i lehtë i aplikacioneve dinamike: React e bën më të lehtë krijimin e aplikacioneve dinamike në internet sepse kërkon më pak kodim dhe ofron më shumë funksionalitet, në krahasim me JavaScript, ku kodimi shpesh bëhet i ndërlikuar shumë shpejt.
- Performanca e përmirësuar: React përdor Virtual DOM, duke krijuar kështu aplikacione në ueb më shpejt. Virtual DOM krahason gjendjet e mëparshme të komponentëve dhe përditëson vetëm artikujt në Real DOM që janë ndryshuar, në vend që të përditësojë përsëri të gjithë komponentët, siç bëjnë aplikacionet e zakonshme të uebit.
- Komponentët e ripërdorshme: Komponentët janë blloqet ndërtuese të çdo aplikacioni React dhe një aplikacion i vetëm zakonisht përbëhet nga disa komponentë. Këta komponentë kanë logjikën dhe kontrollet e tyre dhe ato mund të ripërdoren gjatë gjithë aplikacionit, gjë që redukton në mënyrë dramatike kohën e zhvillimit të aplikacionit.
- Rrjedha e të dhënave me një drejtim: React ndjek një rrjedhë të dhënash me një drejtim. Kjo do të thotë që gjatë dizajnimit të një aplikacioni React, zhvilluesit shpesh vendosin komponentët fëmijë brenda komponentëve prindër. Meqenëse të dhënat rrjedhin në një drejtim të vetëm, bëhet më e lehtë për të korrigjuar gabimet dhe për të ditur se ku ndodh një problem në një aplikacion në momentin në fjalë.
- Kurba e vogël e të mësuarit: React është e lehtë për t'u mësuar, pasi kombinon kryesisht konceptet bazë HTML dhe JavaScript me disa shtesa të dobishme. Megjithatë, siç është rasti me mjetet dhe kornizat e tjera, duhet të kaloni pak kohë për të kuptuar siç duhet librarinë e React.
- Mund të përdoret për zhvillimin e aplikacioneve në ueb dhe celular: Ne tashmë e dimë se React përdoret për zhvillimin e aplikacioneve në ueb, por kjo nuk është e gjitha që mund të bëjë. Ekziston një kornizë e quajtur React Native, që rrjedh nga vetë React, që është jashtëzakonisht e popullarizuar dhe përdoret për krijimin e aplikacioneve të bukura celulare. Pra, në realitet, React mund të përdoret për të krijuar aplikacione në ueb dhe celular.

Arsyet e mësipërme justifikojnë popullaritetin e librarisë React dhe pse ajo po miratohet nga një numër i madh organizatash dhe biznesesh [4].

2.2 React si Single Page Application

Aplikacionet në ueb kanë vuajtur nga interaktiviteti dhe reagimi i dobët ndaj përdoruesve, pavarësisht nga popullariteti i tyre i madh. Ka ende aplikacione në ueb të bazohen në modelin klasik të ndërfaqes me shumë faqe, në të cilin për çdo kërkesë rifreskohet i gjithë aplikacioni. Këto arkitektura kanë shumë kufizime në aspektin e ndërveprimeve njeri-kompjuter. Në një aplikacion tradicional Ueb, sa herë që aplikacioni thërret serverin, serveri jep një faqe të re HTML. Kjo shkakton një rifreskim të faqes në shfletues.

Nëse keni shkruar ndonjëherë një aplikacion në PHP, ky cikël jetësor i faqes duhet të duket i njohur. Në një SPA, pas ngarkimit të faqes së parë, i gjithë ndërveprimi me serverin ndodh përmes thirrjeve AJAX. Këto thirrje AJAX kthejnë të dhëna zakonisht në formatin JSON. Aplikacioni përdor të dhënat JSON për të përditësuar faqen në mënyrë dinamike, pa e ringarkuar faqen. Figura 1 ilustron ndryshimin mes dy qasjeve [5].

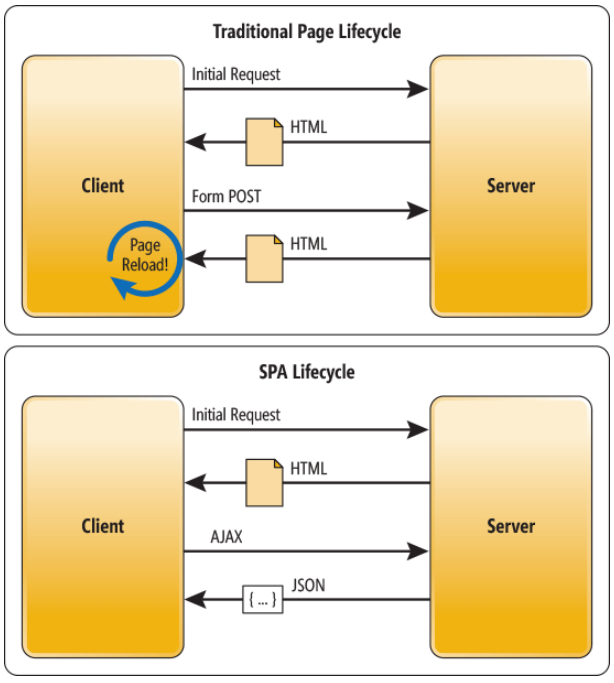

Figura. 1 Cikli jetesor i faqeve tradicionale vs SPA [5]

Një SPA (aplikacion me një faqe) është një implementim i ueb aplikacionit në vetëm një dokument të dhe më pas përditëson përmbajtjen e trupit të atij dokumenti të vetëm nëpërmjet API-ve në JavaScript si XMLHttpRequest dhe Fetch për shfaqjen e përmbajtjeve të ndryshme [6]. Prandaj,

kjo i lejon përdoruesit të përdorin faqet e internetit pa ngarkuar faqe të reja nga serveri, gjë që mund të rezultojë në përfitime të performancës dhe një përvojë më dinamike, me disa disavantazhe të ndërlidhura si SEO.

Me React, zhvilluesit mund të krijojnë komponente të ripërdorshme që mund të përditësohen dhe të paraqiten në mënyrë efikase si përgjigje të ndryshimeve në të dhëna. Ju jeni duke përdorur këtë lloj aplikacionesh çdo ditë. Këto janë, për shembull: Gmail, Google Maps, Facebook ose GitHub. SPA-të kanë të bëjnë me shërbimin e një UX të jashtëzakonshëm duke u përpjekur të imitojnë një mjedis "natyror" në shfletues - pa ringarkime faqesh, pa kohë shtesë pritjeje. Është vetëm një faqe interneti që vizitoni, e cila më pas ngarkon të gjithë përmbajtjen tjetër duke përdorur JavaScript – nga e cila varen shumë. Për më tepër, fleksibiliteti dhe aftësia e React për t'u integruar me libraritë e bëjnë atë një mjet të fuqishëm për ndërtimin e ndërfaqeve dinamike dhe tërheqëse për përdoruesit.

2.2 Krijimi dhe perdorimi i komponentave

Në React, një komponente është një pjesë kodi e ripërdorshme që përfaqëson një pjesë të një ndërfaqeje përdoruesi (user interface). Komponentat na lejojnë të ndajmë UI-në në pjesë të pavarura, të ripërdorshme dhe të mendojmë për secilën pjesë në izolim [7].

Komponentat në React zakonisht deklarohen duke përdorur klasa

```
class Welcome extends React.Component {
    render() {
        return <h1>Hello{this.props.name}</h1>;
    }
}
```

Figura 2. Deklarimi i komponentes duke perdorur klasa

ose funksione të JavaScript [8].

```tsx
function Welcome(props: any) {
    return <h1>Hello, {props.name}</h1>;
}
```

Figura 3. Deklarimi i komponentes duke perdorur funksion

Ato pranojnë të dhëna në formën e "props" (shkurt për properties) dhe kthejnë një përshkrim të ndërfaqes së përdoruesit të komponentit duke përdorur JSX. Komponentat mund të jenë të thjeshta, si një buton. Figura 2 ilustron një shembull.

rateMyTeacherFrontend > src > components > atoms > Button > Button.tsx > ...

```tsx
import React from "react";
import { Button, ButtonProps } from "react-bootstrap";
import styles from './index.module.scss'


interface Props extends ButtonProps {
    title: string;
}

const MyButton = ({ title, ...props }: Props) => {
    return <Button className={styles.custom} {...props}>{title}</Button>
}

export default MyButton
```

Figura 4. Komponente e thjeshte ne React

ose më komplekse, si një formular ose një menu navigimi.

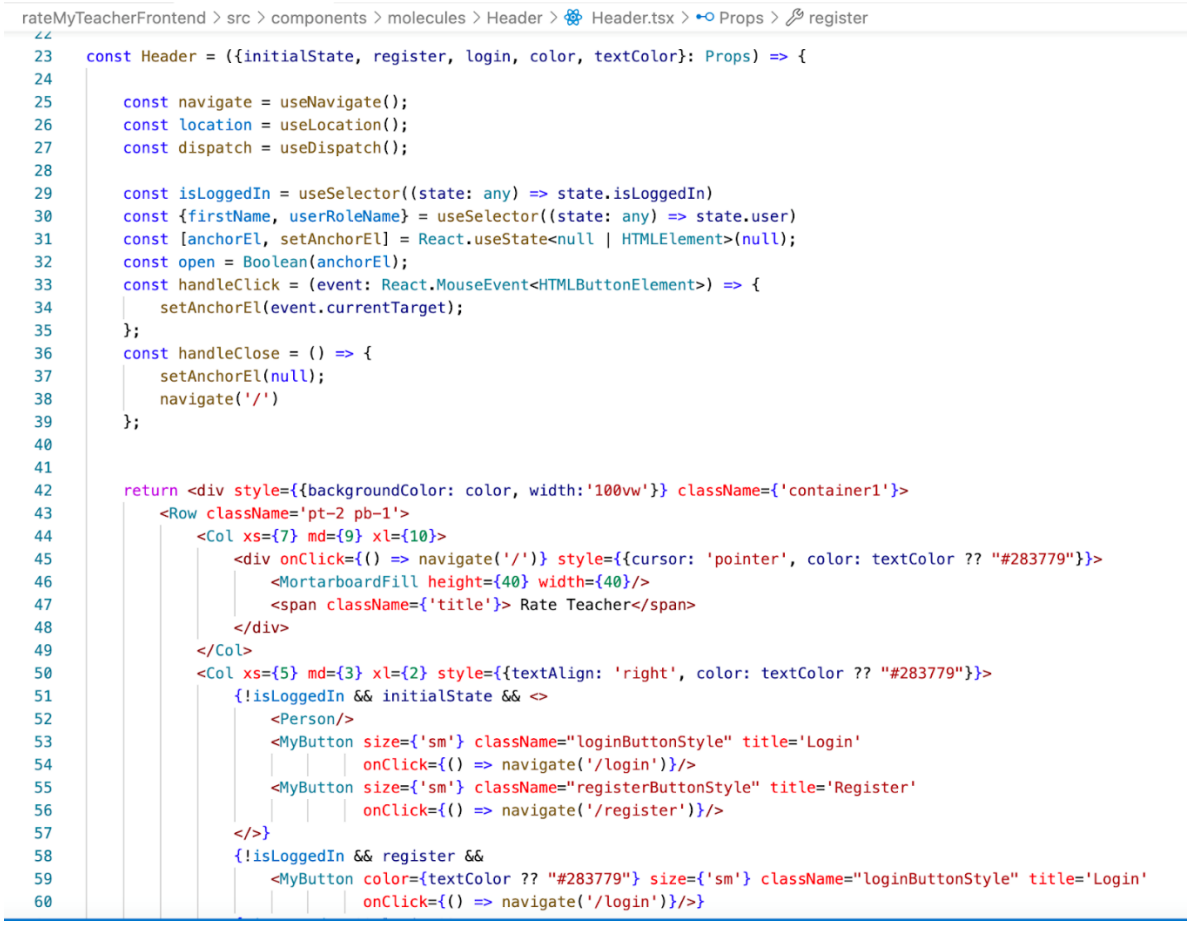

Figura 5. Komponente komplekse ne React

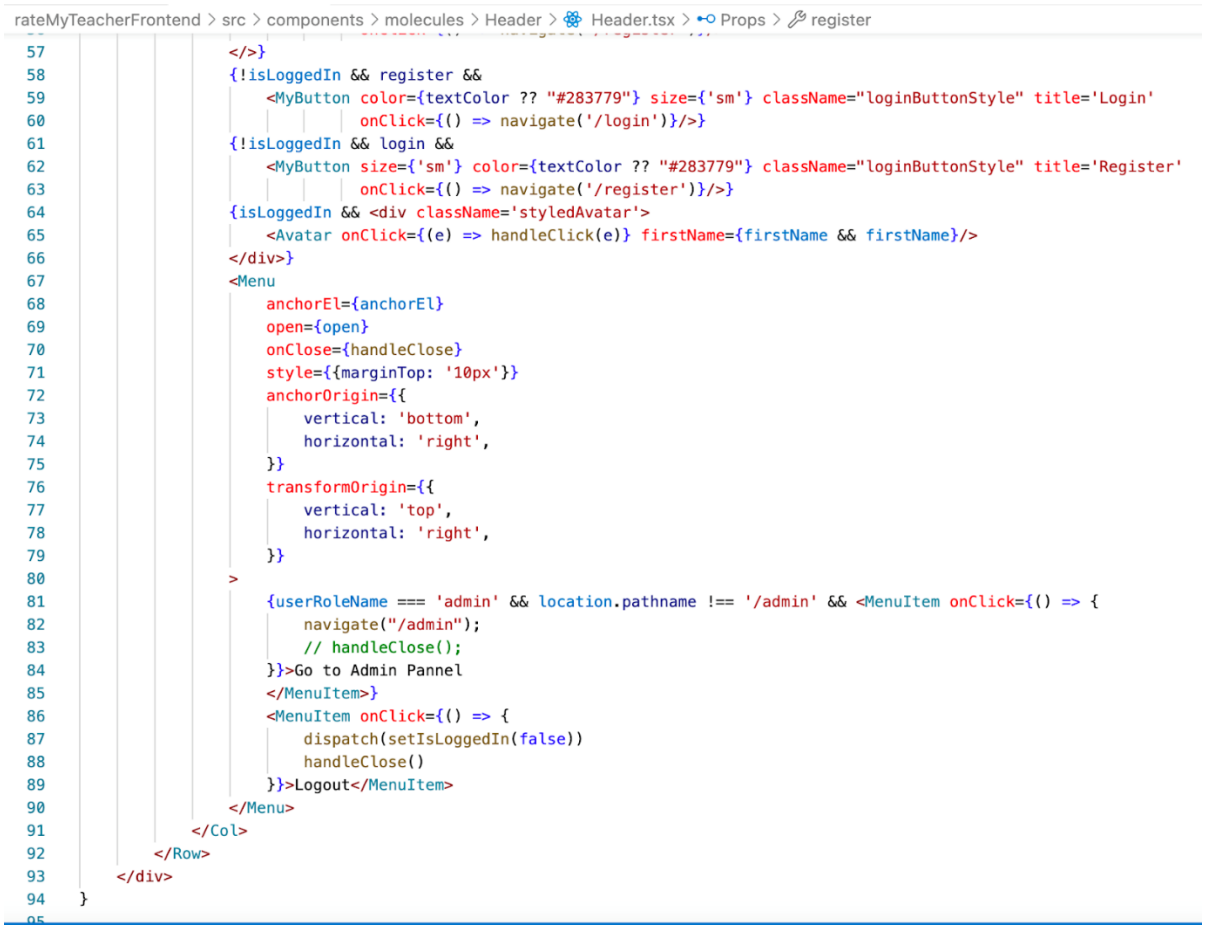

Figura 6. Komponente komplekse ne React

Komponentat gjithashtu mund të futen brenda komponentave të tjera, duke lejuar një strukturë hierarkike në ndërfaqen e përdoruesit. Në të kaluarën, avantazhet që na dhanë komponentat e bazuar në klasa ofronin disa avantazhe të cilat përfshijnë:

- Mundësinë të i'u casen 'gjendjes' së komponentës
- Lifecycle methods si p.sh componentDidMount, componentDidUpdate etj.

Këto ishin më parë shumë të vështira për t'u zbatuar me komponenta të bazuara në funksione. Por me React Hooks, komponentat e bazuara në funksione tani mund të zbatojnë gjithçka që munden komponentët e bazuara në klasa. Dhe jo vetëm kaq, komponentat e bazuara në funksione ofrojnë avantazhe të tjera si:

- Kod më i pastër duke enkapsuluar logjikën

- Përmirësimi i ripërdorimit të kodit duke ndarë logjikën mes komponentë

Gjendja (state) e brendshme e komponentës mund të përdoret gjithashtu për të kontrolluar sjelljen dhe renderimin e saj.

React ofron një sërë metodash të ciklit jetësor (lifecycle methods) që thirren në faza të ndryshme të ciklit jetësor të një komponente, si p.sh. kur renderohet për herë të parë ose kur ndryshojnë props ose gjendja (state) e saj. Kjo i lejon zhvilluesit të shkruajnë kodin që do të funksionojë në pika të veçanta të ciklit jetësor të komponentët, të tilla si marrja e të dhënave ose përditësimi i gjendjes së komponentës. React ofron gjithashtu një mekanizëm të fuqishëm për menaxhimin dhe përditësimin e gjendjes së një komponente, të quajtur gjendje.

Gjendja është një objekt JavaScript që përmban të dhëna që mund të ndryshojnë me kalimin e kohës, të tilla si një vlerë e një fushe hyrëse ose një listë artikujsh. Komponentat në React mund të përdorin gjendjen për të kontrolluar sjelljen dhe renderimin e tyre, dhe React do të përditësojë automatikisht ndërfaqen e përdoruesit kur gjendja të ndryshojë [9].

3. Çka janë React Hooks?

3.1 React Hooks

React Hooks u prezantuan për herë të parë në versionin e React 16.8 në 2019 [10]. Motivi pas prezantimit të Hooks ishte përmirësimi i lexueshmërisë dhe ripërdorimit të kodit duke lejuar zhvilluesit të përdorin gjendjen dhe veçoritë e tjera të React në komponentët e bazuara mbi funksione.

Përpara Hooks, 'state' dhe veçoritë e tjera mund të përdoreshin vetëm në komponentët e bazuara mbi klasë, gjë që e bënte më të vështirë ndarjen e logjikës midis komponentëve dhe e bënte më të vështirë testimin, kuptimin dhe arsyetimin rreth sjelljes së kodit. Përpara propozimit të Hooks, kishte disa mënyra për të arritur funksionalitete të ngjashme, të tilla si përdorimi i komponentëve të rendit më të lartë (HOC) dhe render props, por këto modele kishin disa anë negative, si duke e bërë kodin më kompleks dhe më të vështirë për t'u lexuar.

Hooks janë një shtesë më e fundit e React dhe janë bërë shpejt të njohura në mesin e zhvilluesve të React pasi u lejojnë zhvilluesve të shkruajnë komponentë funksionalë dhe të kompozueshëm me të njëjtat aftësi si komponentët e klasës. Facebook ka vazhduar të zhvillojë dhe përmirësojë Hooks që nga prezantimi i tyre në 2019, dhe tani ato janë një pjesë integrale e librarisë React. React Hooks janë funksione që lejojnë zhvilluesit të përdorin gjendjen dhe veçoritë e tjera të React në komponentet e bazuara mbi funksione, në vend që të përdorin komponentët e bazuara mbi klasë. Përpara React Hooks, 'state' dhe veçori të tjera si 'lifecycle methods' mund të përdoreshin vetëm në komponentët e klasës.

Me prezantimin e Hooks, zhvilluesit tani mund të përdorin 'state' dhe veçori të tjera edhe në komponentët funksionalë, duke bërë të mundur shkrimin e komponentëve funksionalë, të ripërdorshëm dhe të kompozueshëm. Hooks u shfaqen për të zgjidhur disa nga problemet që lidhen me përdorimin e komponentëve e bazuara mbi klasë në React. Megjithatë, disa nga këto probleme nuk janë të lidhura drejtpërdrejt me React, por, përkundrazi, me mënyrën se si janë të dizajnuara klasat në JavaScript.

Disa nga këto probleme janë:

- Autobinding dhe përdorimi i 'this'

```
16  import React, { Component } from "react";
17
18  class Card extends Component {
19    constructor(props) {
20      super(props);
21      this.state = { name: "John Doe" };
22      this.changeName = this.changeName.bind(this);
23    }
24
25    changeName() {
26      this.setState({ name: "Jane Doe" });
27    }
28
29    render() {
30      return (
31        <div>
32          <p>My name is {this.state.name}.</p>
33          <button onClick={this.changeName}>Change Name</button>
34        </div>
35      );
36    }
```

Figura 7. Autobinding dhe perdorimi i 'this'

Ky kod tregon një komponent të thjeshtë të klasës që renderon një ‘state’ të emrit në UI dhe ofron një buton për të ndryshuar emrin. Siç e shihni, duhet në mënyrë eksplicite të ‘bind this’, të thërrisni super(props) në konstruktorin bazë dhe gjithmonë të parashtesoni gjendjen ose metodat tuaja me ‘this’ për t'iu casur atyre. Këto detaje të vogla janë nga mënyra se si janë dizajnuar klasat ES6 dhe janë disa nga shkaqet e zakonshme të gabimeve në aplikacionet React.

- Verbose syntax

  Komponentët e klasës në React kanë një sintaksë të hollësishme ‘verbose syntax’që shpesh mund të rezultojë në "komponentë shumë të mëdhenj" - komponentë me shumë logjikë të ndarë në disa ‘lifecycle methods’ që janë të vështira për t'u ndjekur.

  Gjithashtu, API e ‘lifecycle methods’ ju detyron të përsërisni logjikën e lidhur nëpër metoda të ndryshme të ciklit jetësor përgjatë komponentit.

- Vështirë për t’u ripërdorur dhe për të ndarë logjiken

Duke përdorur komponentët e bazuara mbi klasë në React, shpesh ju nevojiten modele të ndërlikuara si modeli i komponentëve të rendit më të lartë (HOC) që e bëjnë kodin tuaj të vështirë për t'u lexuar dhe mirëmbajtur. HOC-të përdoren shpesh për t’i adresuar cështjet si autorizimi, logging dhe marrja e të dhënave; dmth detyrat që përfshijnë të gjithë aplikacionin dhe përndryshe do të çonin në logjikë të përsëritur.

Problemet lindin kur përpiqeni të përdorni shumë HOC si p.sh withRouter, withAuth, withTheme, withLogger etj. Kjo dërgon drejt kodit i cili bëhet vështirë i lexueshëm, sic mund ta shihni më poshtë:

```
import React from "react";
import WithRouter from “./components/WithRouter";
import WithAuth from "./components/WithRouter";
import WithLogger from "./components/WithRouter";
import WithTheme from “./components/WithRouter";

const SomeComponent = (props) => {
    return (
    // some jsx
    )
}
export default WithTheme(
    WithAuth(
        WithLogger(
            WithRouter(SomeComponent);
    )
  )
);
```

Figura 8. Perdorimi i HOC

HOC mund të rezultojnë në një strukturë ‘deeply nested’ e cila bën shumë të vështirë të debugojmë aplikacionin tonë [11].

3.2 Praktikat më të mira të përdorimit të React Hooks

3.2.1 Rregullat e React Hooks

Dy rregulla të rëndësishme duhet të ndiqen kur punoni me Hooks. Këto rregulla janë thelbësore për të ruajtur rendin, për të shmangur gabimet e panevojshme dhe për të na ndihmuar të shkruajmë kod të pastër. Këto rregulla janë [12]:

1. Thërrisni Hooks vetëm në nivelin më të lartë

   Niveli i lartë i një komponentës së bazuar mbi funksion është baza e trupit të funksionit përpara se të ktheni JSX elementet. Këtu mund t'i thërrisni të gjitha Hook-et tuaja, kështu që React mund të ruajë dhe të mbajë shënim rendin në të cilin thirren këto Hooks.

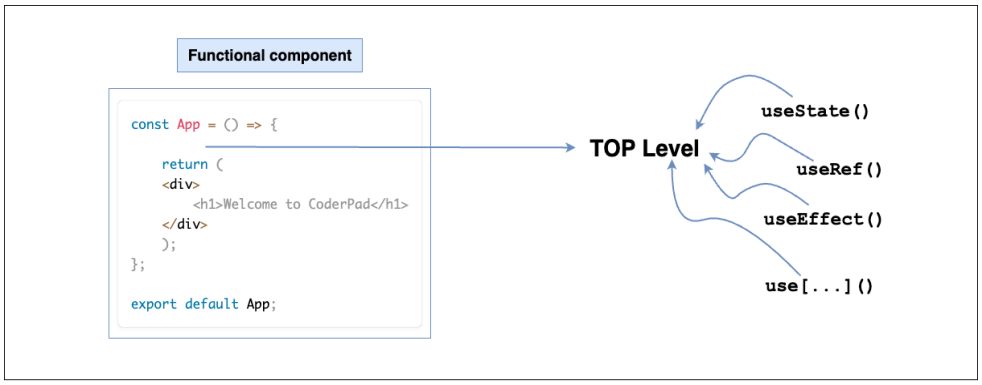

Figura 9 Thirrja e Hooks ne nivelin me te larte [12]

Thirrja e Hooks tuaj në krye të funksionit do të thotë që nuk duhet t'i thërrisni ato brenda loops, conditions apo nested functions .

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

const App = () => {
    // ✅ - correct
    const [todos, setTodos] = useState([]);

    // ❌ - Breaks the call order
    if (todos) {
        const [count, setSetcount] = useState(todos.length);
    }

    // ❌ - Breaks the call order
    todos.forEach(todo => {
        useEffect(() => {
            console.log(todos);
        });
    });

    // ✅ - correct
    useEffect(() => {
        console.log(todos);
    });

    return (
        // ...
    );
};

export default App;
```

Figura 10. Menyra korrekte dhe e gabuar per t'i thirrur Hooks [12]

Është e nevojshme t'i thërrisni Hooks në nivelin më të lartë të komponentit tuaj funksional për të siguruar që sa herë që komponenti juaj renderohet, këto Hooks thirren në të njëjtin rend. Rendi i thirrjes është thelbësor që Hooks të funksionojnë siç duhet.

Për shembull, kur merrni parasysh kodin e mësipërm, ai përdor vlerën e gjendjes todos për të vendosur nëse do të thërrasë Hook-in e dytë dhe të tretë. Kur kushti dështon, ose ndodh diçka me ciklin, ai prish rendin. React atëherë do ta ketë të vështirë të kuptojë se si të ruajë gjendjen e komponentit tuaj. Kjo është ajo që bën React kur përdorni Hooks. Ai identifikon rendin në të cilin Hooks përdoren në paraqitjen fillestare, më pas në paraqitjet pasuese, React do të jetë në gjendje të ruajë gjendjen e komponentes tuaj.

2. Thërrisni Hooks vetëm nga komponentet funksione ose Hooks të tjerë

   Një rregull i dytë është që Hooks duhet të përdoren dhe thirren vetëm në komponentët funksione të React ose Hooks të personalizuara, jo funksionet e rregullta JavaScript ose komponentët e klasës React.

Për të krijuar një Hook të personalizuar, thjesht krijoni një funksion JavaScript ku emri i funksionit fillon me 'use'. Pastaj, mund ta përdorni për të thirrur Hooks të tjerë. Për shembull, këtu është një zbatim i Hook custom useMyName:

```
export default function useMyName(name) {
    const [state, setState] = useState(value);

    useEffect(() => {
    // ...
    });

    return anyThing;
}
```

Figura 11 Hook i personalizuar [12]

3.2.2 ESLint Plugin per React Hooks

Plugin ESLint për React Hooks është një mjet që i ndihmon zhvilluesit të kapin gabimet e mundshme dhe të ndjekin praktikat më të mira kur përdorin React Hooks në kodin e tyre. Ky plugin ofron një sërë rregullash që mund të shtohen në një file konfigurimi ESLint për të kontrolluar automatikisht për probleme me përdorimin e Hooks [13].

Ky plugin përfshihet ‘by default’ në Create React App por eslint-plugin-react-hooks mund dhe të instalohet duke përdorur npm apo yarn dhe më pas të shtohet në file tuaj të konfigurimit të ESLint [14].

Pasi të shtohet, mund të aktivizoni rregullat e ofruara nga plugin-i. Një file konfigurimi i ESLint duket si në vijim:

```
// Your ESLint configuration
{
    "plugins": [
    // ...
    "react-hooks"
    ],
    "rules": {
    // ...
    "react-hooks/rules-of-hooks": "error", // Checks rules of Hooks
    "react-hooks/exhaustive-deps": "warn" // Checks effect dependencies
    }
}
```

Figura 12 Nje file konfigurimi ESLint [14]

Disa nga rregullat e ofruara nga plugin-i përfshijnë:

- Sigurimi që Hooks thirren vetëm në nivelin më të lartë të një komponenteje ose në Hooks të personalizuara
- Verifikimi që Hooks thirren vetëm nga komponentet funksione apo Hooks të tjerë
- Kontrollimi i renditjes së thirrjes së Hooks për t’u siguruar që ato thirren në të njejtin rend në cdo renderim.
- Paralajmërim kundër përdorimit të ‘state’ ose useEffect Hooks brenda loops apo kushteve.

ESLint gjithëmonë paralajmëron për mos ndjekjen e këtyre rregullave, dhe paralajmërimi duket si në vijim:

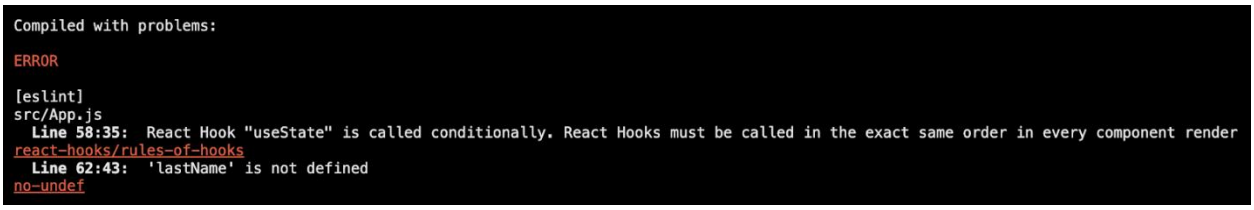

Figura 13. Paralajmerimi per mos ndjekjen e rregullave te ESLint [12]

Duke përdorur këtë plugin, ju mund të kapni gabimet herët dhe të shmangni gabimet e mundshme dhe sjelljen e papritur në kodin tuaj.

3.2.3 Praktikat më të mira të përdorimit të React Hooks

Disa nga praktikat më të mira që mund t'i ndjekim kur përdorim React Hooks janë:

- Thjeshtësoni Hooks tuaj

  Mbajtja e thjeshtë e React Hooks do t'ju japë fuqinë për të kontrolluar dhe manipuluar në mënyrë efektive atë që ndodh në një komponente gjatë gjithë jetës së saj. Shmangni sa më shumë që të jetë e mundur të shkruani hooks të personalizuar.

  Nëse gjeni veten duke përdorur një grup hooks të personalizuar, mund të krijoni një hook të personalizuar që vepron si një mbështjellës për këto.

- Organizoni dhe strukturoni Hooks tuaj

  Një nga avantazhet e React Hooks është aftësia për të shkruar më pak kod që lexohet lehtë. Në disa raste, perdorimi i shpeshte i useEffect() dhe useState() mund të shkaktoj konfusion. Kur e mbani komponentin tuaj të organizuar, kjo do të ndihmojë në lexueshmërinë dhe do të mbajë rrjedhën e komponentave tuaja të qëndrueshme dhe të parashikueshme [14].

- Përdorni React Hooks snippets

  React Hooks Snippets është një extenstion në Visual Studio Code për t'i bërë React Hooks më të lehtë dhe më të shpejtë per tu perdorur. Aktualisht, mbështeten pesë hooks: useEffect, useState, useContext, useCallback, useMemo [14].

- Respekto rregullat e React Hooks

  Përpiquni të mbani gjithmonë në konsideratë dy rregullat e Hooks që cekëm më herët gjatë punës me React Hooks. Thërrisni hooks tuaj vetëm në nivelin më të lartë. Mos i thirrni hooks brenda loops, kushteve ose funksione të ndërlidhura. Thirrni gjithmonë Hooks nga komponentët e funksionit React ose nga hooks të personalizuara, mos thirrni hooks nga funksionet e rregullta JavaScript. Plugin i ESlint i quajtur eslint-plugin-react-hooks zbaton këto dy rregulla, ju mund ta shtoni këtë plugin në projektin tuaj [14].

Vlen të ceket që praktikat më të mira nuk janë zgjidhur plotësisht sepse hooks janë ende relativisht të reja.

3.3 Hooks më të popullarizuara

Sic e cekëm më herët para versionit 16.8 të React, zhvilluesit detyroheshin të shkruanin komponente të bazuara mbi klasë për t'i perdorur disa vecori të React-it. Por tani kjo mundësohet nga React Hooks dhe për komponentet e bazuara mbi funksion, dhe në këtë pjesë të punimit do të flasim shkurtimisht rreth Hooks më të popullarizuara dhe si funksionojnë ato.

- useState

  Është hook më i rëndësishëm dhe më i përdoruri. Cdo e dhënë që ndryshon në aplikacion quhet gjendje (state), kur ndonjë nga të dhënat ndryshon, React e ri-rendoron ndërfaqen (UI). Pra, ky hook na lejon të shtojmë 'gjendjen' në komponentet tona funksione. Kthen një 'array' me dy elemente: gjendjen aktuale dhe një funksion për të përditësuar gjendjen [15] . Detaje më të hollësishme rreth këtij hook, dhe si e kemi përdorur atë në aplikacionin 'Rate my Teacher' do t'i shtjellojmë në pjesën në vijim të punimit.

```
import { useState } from 'react'
const [state, updaterFn] = useState('');
```

Figura 14. UseState

- useEffect

  useEffect është një tjetër hook i rëndësishem i React që përdoret në shumicën e projekteve. Bën një gjë të ngjashme me 'lifecycle methods' componentDidMount, componentWillUnmount dhe componentDidUpdate të komponentes të bazuar në klasë. useEffect na ofron një mundësi për të shkruar kod imperativ që mund të kenë efekte anësore në aplikacion, të tilla si marrja e të dhënave ose manipulimi i DOM. Thirret pasi komponentja të jetë renderohet dhe mund të përdoret gjithashtu për të pastruar çdo efekt përpara se komponenteja të bëhet 'unmount' [15].

```
import { useState, useEffect } from 'react'
const App = () => {
    const [count, setCount] = useState(0)
    useEffect(() => {
        console.log(count)
    })
    return (<div>... </div>
    )
```

Figura 15. UseEffect

Në kodin e mësipërm, thjeshtë kemi log variablen count në useEffect. Kjo do të thirret pas çdo renderimi të komponentës.

- useContext

  React Context API ofron një mënyrë për të ndarë gjendjen ose të dhënat në të gjithë pemën e komponentes ne React. API ka qenë i disponueshëm në React, si një veçori eksperimentale, për një kohë, por u bë i sigurt për t'u përdorur në React 16.3.0. API e bën të lehtë ndarjen e të dhënave midis komponentëve duke eliminuar 'prop drilling'. Ky hook na lejon të punojmë me React's Context API. Ofron një mundësi për të ndarë 'gjendjen' mes komponenteve pa kaluar 'props' manualisht [16].

- useRef

  Ju lejon të krijoni një objekt 'ref' dhe ta lidhni atë me një element ose një instance te komponentes. Mund të përdoret për të ruajtur vlerat ose referenca të ndryshueshme që nuk shkaktojnë ri-renderim. Rasti i zakonshëm i përdorimit të kësaj, është casje ne elementet HTML nga DOM [16].

```
function App() {
    const myBtn = React.useRef(null);
    const handleBtn = () => myBtn.current.click();
    return (<button ref={myBtn} onChange={handleBtn}></button> )}
```

Figura 16. UseRef

- useReducer

  Është shumë e ngjashme me setState, është një mënyrë tjetër për të menaxhuar gjendjen

duke përdorur Redux pattern. Në vend që të përditësojmë drejtpërdrejt gjendjen, ne dërgojmë veprime që shkojnë në një funksion 'reducer' dhe ky funksion zbulon se si të llogarisim gjendjen tjetër [15].

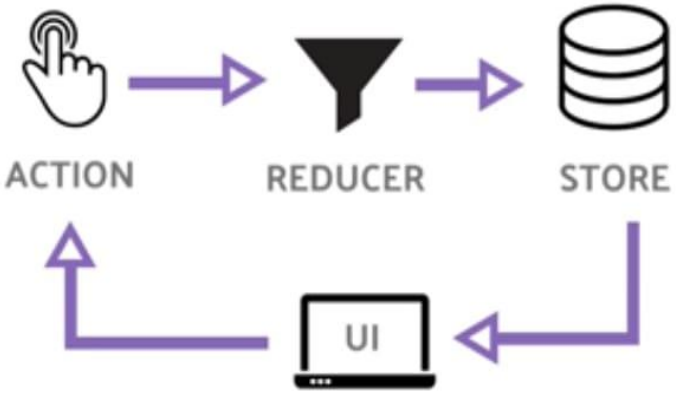

Figura 17. Arkitektura e useReducer [15]

- useMemo

  Kjo ju lejon të memorizoni një vlerë dhe të parandaloni ri-renderimet e panevojshme. Ai kthen një version të memoizuar të vlerës që ndryshon vetëm nëse një nga varësitë e specifikuara ka ndryshuar. Ky hook do t'ju ndihmojë të optimizoni koston llogaritëse ose të përmirësoni performancën. Përdoret kryesisht kur na nevojiten për të bërë llogaritje të shtrenjta [15].

```
function useMemo() {

    const [count, setCount] = React.useState(60);
    const expensiveCount = useMemo(() => {
        return count ** 2;
    }, [count]) // recompute when count changes.
}
```

Figura 18. UseMemo

- useCallback

  Kjo ju lejon të memorizoni një funksion dhe të parandaloni ri-renderimet e panevojshme. Ai kthen një version të memorizuar të funksionit callback që ndryshon vetëm nëse një nga varësitë e specifikuara ka ndryshuar [15].

```
function useCallbackDemo() {
    const [count, setCount] = useState(60);
    const showCount = React.useCallback(() => {
        alert(`Count ${count}`);
    }, [count])
    return <> <SomeChild handler={showCount} /></>
}
```

Figura 19. UseCallback

Këto janë hooks më të përdorura, por ka shumë më tepër hooks në dispozicion në varësi të rastit të përdorimit.

3.4 Benefitet e React Hooks

Siç e kemi parë tashmë, React Hooks e bën jetën tonë më të lehtë në shumë mënyra, disa nga benefitet do t'i listojmë më poshtë:

- Përmirësimi i ripërdorimit të kodit:
  Hooks lejojnë komponentët funksionalë të përdorin gjendjen dhe veçori të tjera, gjë që e bën më të lehtë ndarjen e logjikës midis komponentëve [17].
- Përmirësimi i lexueshmërisë së kodit:
  Hooks e bëjnë më të lehtë kuptimin e sjelljes së një komponenti duke mbajtur logjikën në një vend të vetëm dhe duke shmangur nevojën për klasa dhe metoda të ciklit jetësor. Përveç që kodi është më i lehtë për t'u lexuar, është gjithashtu shumë më e lehtë të lexohet pema e komponentes në mjetet e React dev gjatë debugim-it [17].
- Përmirësimi i performancës:
  Hooks u lejojnë zhvilluesve të ri-kthejnë vetëm pjesët e një komponenti që kanë ndryshuar, gjë që mund të përmirësojë performancën në krahasim me rikthimin e të gjithë komponentit [18].
- Testim më i mirë:
  Hooks e bëjnë më të lehtë testimin e komponentëve të veçuar, pasi ato nuk mbështeten në klasa dhe metoda të ciklit jetësor [18].
- Lehtë për t'u mësuar:

Hooks janë të lehta për t'u mësuar për zhvilluesit që janë tashmë të njohur me React dhe nuk kërkojnë ndonjë koncept ose sintaksë shtesë për t'u kuptuar.

- Lehtë për t'u korrigjuar:
  Hooks lejojnë zhvilluesit të shtojnë një etiketë në një Hook të personalizuar për qëllime korrigjimi.
- Kontroll më i mirë i sjelljes së komponentit:
  Hooks japin më shumë fleksibilitet për të kontrolluar sjelljen e komponentit, ata lejojnë zhvilluesit të kontrollojnë kur një komponent duhet të përditësohet dhe kur jo.

Si përmbledhje, Hooks e bën më të lehtë shkrimin e komponentëve funksionalë dhe të kompozueshëm, përmirëson performancën dhe e bën më të lehtë testimin dhe korrigjimin e kodit.

4. Krijimi i një aplikacioni duke përdorur React Hooks

4.1 Teknologjitë e përdorura

Për zhvillimin e aplikacionit 'Rate Teacher' kemi përdorur këto teknologji:
Për zhvillim në frontend kemi përdorur ReactJS, detaje më të hollësishme rreth cka është ReactJS dhe benefiteve që ofron janë diskutuar në pjesën e parë të punimit.
Për zhvillim në backend kemi përdorur NestJS, Nest.js është një nga framework-at e Node.js me rritjen më të shpejtë për ndërtimin e aplikacioneve të backend duke përdorur Node.js. Është i njohur për zhvillimin e aplikacioneve lehtë të testueshme,lehtë të mirëmbahen duke përdorur JavaScript dhe TypeScript moderne.

Meqenëse ky framework përdor TypeScript, Nest.js është veçanërisht i popullarizuar ndër ekipet që kërkojnë të përdorin fuqinë e kontrollit të tipeve të TypeScript. Për më tepër, është e thjeshtë për t'u mësuar dhe zotëruar, me një CLI të fuqishëm për të rritur produktivitetin dhe lehtësinë e zhvillimit. Ky CLI i fuqishëm e bën të lehtë të nisësh çdo projekt nga ana e serverit dhe ta shohësh atë deri në përfundim.

Për më tepër, Nest.js mban dokumentacion të detajuar dhe komuniteti i zhvilluesve dhe kontribuesve të tij është shumë aktiv dhe i gatshëm për t'iu përgjigjur çështjeve [19].

Për ruajten e të dhënave është përdorur Mongo DB, Mongo DB është një databazë jo relacionale e cila ofron fleksibilitet në qasje të të dhënave, ruan të dhënat në një lloj formati JSON të quajtur BSON. Ndryshe nga databazat relacionale të ndërtuara me tabela, mongo DB ndërtohet mbi formatin e të dhënave JSON me të cilin krijohen skemat. Rekordet në një bazë të dhënash MongoDB quhen dokumente dhe vlerat e fushës mund të përfshijnë numra, vargje, booleans, vargje apo edhe dokumente 'nested' [20].

Editori i kodit që kemi përdorur është Visual Studio Code, ky editor ofron mbështetje gjatë zhvillimit për debugging, 'version control' etj.

4.2 Skema e databazës

Sic e potencuam më herët për ruajtjen e të dhënave kemi përdorur Mongo DB. Skema e databazës duket si në vijim:

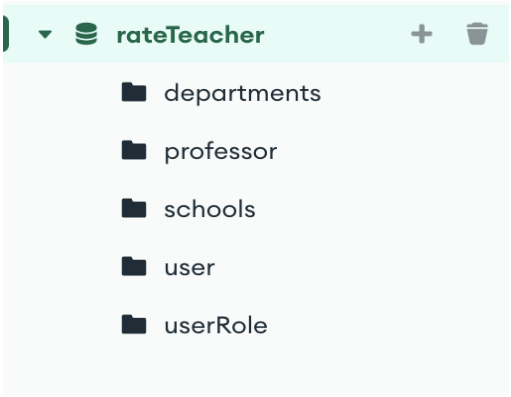

Figura 20. Skema e databazes

Pra, kemi këto tabela:

- Departments, për ruajten e departamenteve
- Professor, për ruajtjen e profesorëve
- Schools, për ruajtjen e institucioneve arsimore
- User, për t'i ruajtur përdoruesit të cilët regjistrohen në platformë
- UserRole, për ruajtjen e rolit të përdoruesit, admin apo përdorues

Një rekord në tabelën departments, duket si vijon:

```
_id: ObjectId('63302894fa7424fe6311fad0')
departmentName: "Computer science"
__v: 0
```

Figura 21. Nje rekord i tabeles Departments

Ky rekord përmban id e cila është e autogjeneruar dhe emrin e departamentit në këtë rast 'Computer Science'.

Një rekord i tabelës professor, duket si vijon:

```
_id: ObjectId('6332e94f437a8e2591773b5e')
professorName: "alba"
schoolName: "new"
schoolId: "6332e943437a8e2591773b52"
▾ departments: Array
  ▾ 0: Object
      label: "Computer science"
      value: "63302894fa7424fe6311fad0"
  ▾ 1: Object
      label: "new dep"
      value: "6332e92b437a8e2591773b4b"
▸ ratings: Array
  __v: 0
```


Figura 22. Nje rekord ne tabelen Professor

Ky rekord përmban fushat si id-në e autogjeneruar, emrin e profesorit, emrin dhe id-në e shkollës si pasojë e lidhjes mes këtyre dy tabelave, departments një vektor (array) me id-të e departamenteve që i përket ky profesor po ashtu dhe një vektor me vlerësimet për këtë profesor.

Një rekord i tabelës schools, duket si vijon:


```
▸ _id: ObjectId('633028d1fa7424fe6311faee')
  schoolName: "UMIB"
  schoolZip: "4000"
  schoolLocation: "Mitrovice"
▸ departments: Array
  __v: 0
```


Figura 23. Nje rekord ne tabelen Schools

Ky rekord përmban fushat si id-në e autogjeneruar, emrin e shkollës, zip code, lokacionin e shkollës, dhe një vektor që ruan departamentet e kësaj shkolle.

Një rekord i tabelës user, duket si vijon:


```
▸ _id: ObjectId('63d17e0a14c93fbee5f145dc')
  email: "diellzapllana@gmail.com"
  password: "$2b$10$Vfo3QS8ZKeDAvLgt/E5IkO3hxttv8GYeK0t5.5sTL38kR75kVIGm."
  firstName: "Diellza1"
  lastName: "Pllana"
  userRoleName: "user"
  __v: 0
```


Figura 24. Nje rekord ne tabelen User

Ky rekord përmban fushat si id-në e autogjeneruar, email, password, emrin dhe mbiemrin dhe rolin e përdoruesit në aplikacion.

4.3 Përshkrimi i aplikacionit

Aplikacioni mundëson vlerësimin e performancës së profesorëve dhe kualitetin e mësimdhënies së tij/saj duke u bazuar në disa pyetje. Përdoruesit mund dhe të japin komente rreth performancës së profesorëve. Aplikacioni ka dy module kryesore:

- Moduli i admin-it
- Moduli i përdoruesit

Pasi përdoruesi të regjistrohet në sistem 'by default' ka rolin user, nëse deshirojmë që këtij useri t'i japim rolin admin, këtë mund ta bëjë dikush që ka rolin admin, përmes kontroll panelit të aplikacionit të ndryshojë rolin e user-it specifik. Më poshtë mund të shihni se si duket forma e regjistrimit.

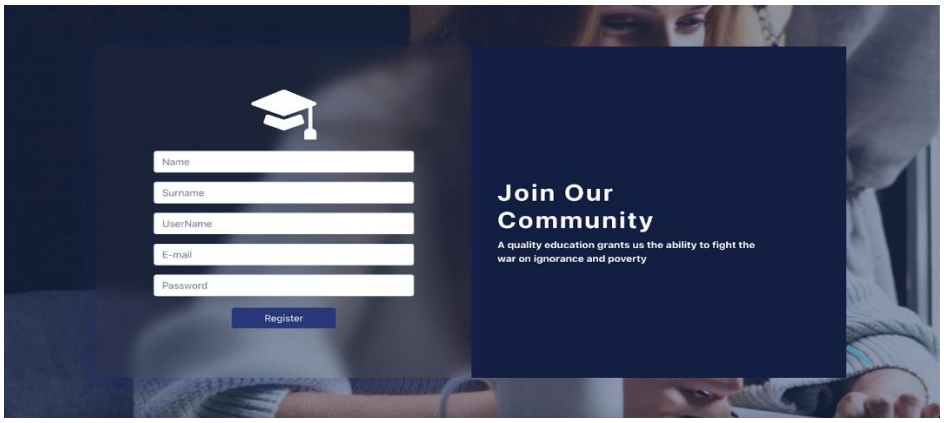

Figura 25. Forma per regjistrim ne aplikacion

Forma e regjistrimit përmban këto inpute : emri, mbiemri, username, email dhe password. Për t'u regjistruar bëhet validimi i të dhënave, ky validim bëhet në këtë pjesë të kodit:

```
const _register = (data: any) => {

    if (data?.firstName && data?.lastName && data?.userName && data?.password && data?.email) {
        if (data?.email.match(/^[a-zA-Z0-9.! #$%&'*+/=? ^_`{|}~-]+@[a-zA-Z0-9-]+(?:\. [a-zA-Z0-9-]+)*$/)) {
            toast.warning("Invalid email")
        } else if (data?.password.length < 6) {
            toast.warning("Password must have more than 6 characters")
        } else if (data?.firstName && data?.lastName && data?.userName && data?.password && data?.email) {
            register(data).then((res: any) => {
                console.log(res, 'res.response?.data?.message ')
                if (res.response?.data?.message === "Email has been taken") {
                    toast.warning("Email already exists")
                } else if (res?.statusText === "Created") {
                    toast.success("You registered successfuly")
                }
            })
        }
    } else {
        toast.warning("All fields are mandatory!")
    }
}
```

Figura 26. Validimi i te dhenave ne regjistrim

Fillimisht kushtëzimi i parë kontroll nëse të gjitha fushat janë plotësuar nëse jo, atëherë shfaqet ky mesazh paralajmërimi:

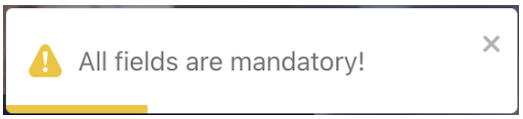

Figura 27. Mesazh paralajmerimi per mosplotesimin e te gjitha fushave

Pastaj kontrollohet me një regex nëse email-i nuk është valid, pas asaj kontrollohet nëse fjalëkalimi ka më pak se 6 karaktere, nëse po atëherë shfaqet ky mesazh paralajmërimi:

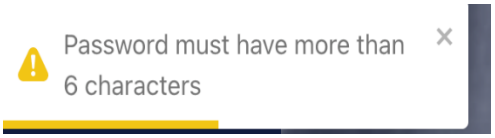

Figura 28. Mesazh paralajmerimi per gjatesine e fjalekalimit

Pastaj kontrollohet nëse kjo email ekziston në sistem, nëse po atëherë shfaqet ky mesazh paralajmërues:

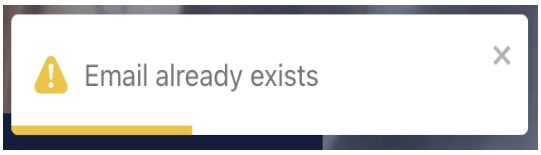

Figura 29. Mesazh paralajmerimi per email ekzistues

Pas regjistrimit përdoruesi mund të logohet në sistem përmes kësaj forme të login-it:

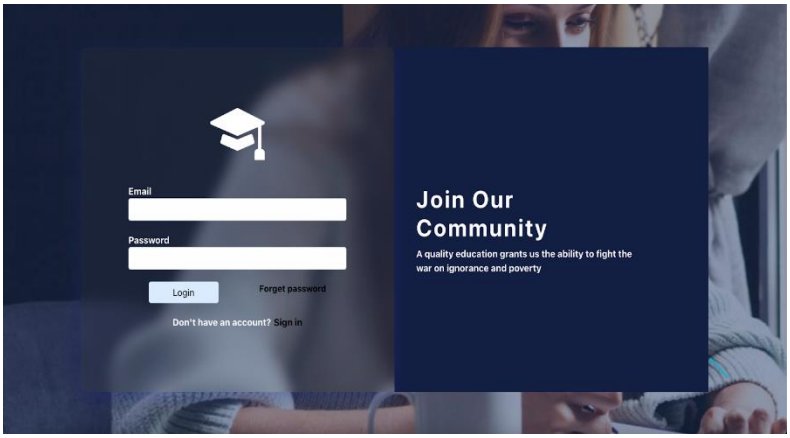

Figura 30. Forma per login

Ngjajshëm si për regjistrim, dhe në formën për login ekzistojnë validime: kur email-i nuk është valid, ndonjë prej fushave nuk është plotësuar, kur fjalëkalimi është më i shkurtër se 6 karaktere.

Pasi të logohemi shfaqet kryefaqja, me një input autocomplete, ku shkruajmë emrin e profesorit faqen e së cilit dëshirojmë të shohim. Kryefaqja duket si në vijim:

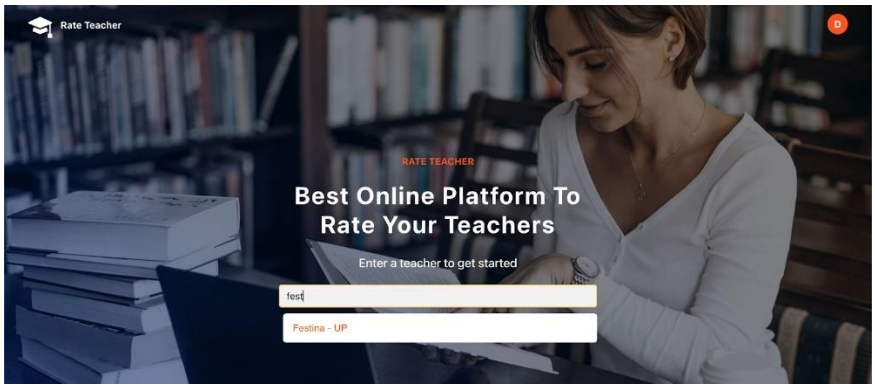

Figura 31. Kryefaqja e aplikacionit

Pasi të klikojmë në njërën nga opsionet e shfaqura në dropdown atëherë shfaqet faqja e profesorit përkatës.

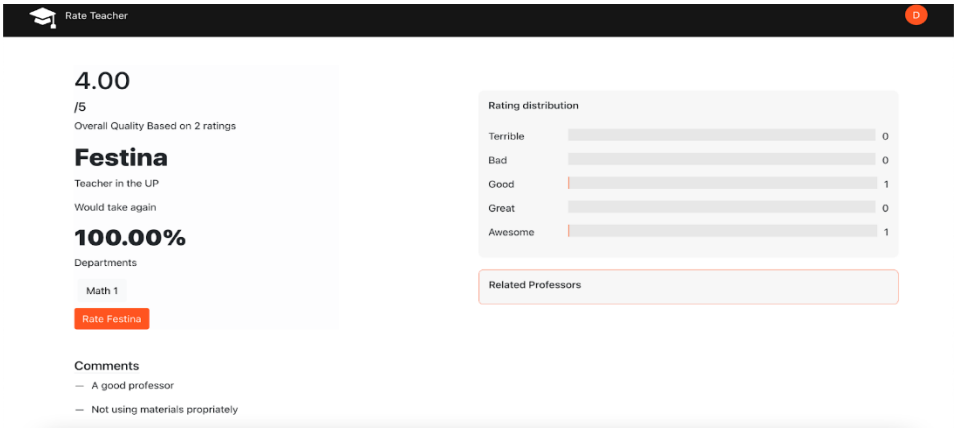

Figura 32. Faqja e profesorit

Në të djathtën e faqes paraqiten pesë shirita te progesit, shkallet e tyre janë si më poshtë: terrible, bad, good, great, awesome dhe në fund të tyre janë numri i votave që rritet proporcionalisht me ngjyrën portokalli. Në anën e majtë të faqes shfaqet vlerësimi i përgjithshëm që në rastin e figurës 23 është 4/5 duke u bazuar në 2 vlerësimet që ka marrë kjo profesoreshë.

Pyetjes nëse do të zgjidhit prapë këtë profesoreshë, të gjithë i'u janë përgjirur me 'po' prandaj shohim përqindjen 100%. Në fund shohim dhe seksionin e komenteve mbi profesorin.

Nëse klikojmë mbi butonin 'Rate Festina' atëherë shfaqet faqja e rankimit e cila duket si vijon:

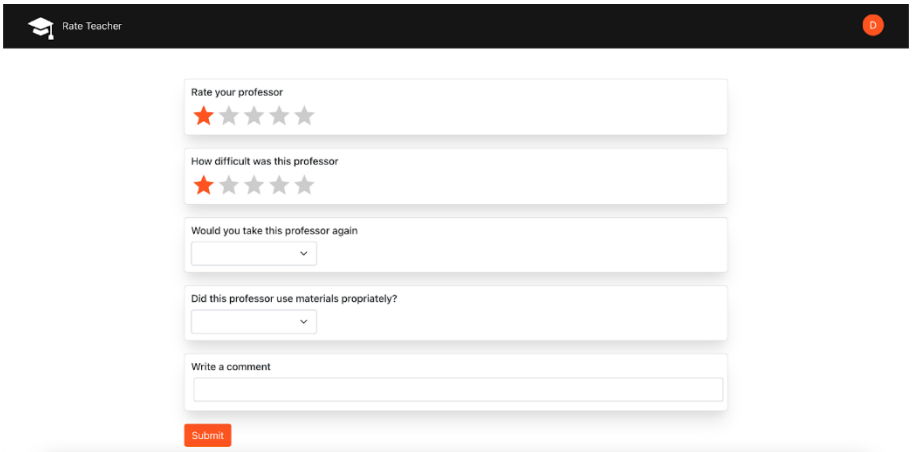

Figura 33. Faqja e rankimit

Në figurën e mësipërme shihet pyetësori që përdoruesi duhet të plotësojë në menyrë që të japë vlerësimin mbi profesorin. Të gjitha fushat janë të detyrueshme për plotësim dhe nëse njëra nuk plotesohet, përdoruesi ndalohet nga mesazhet paralajmëruese të dorëzojë pyetësorin.

Nëse përdoruesi ka rolin admin atëherë ka qasje dhe në kontroll panelin i cili duket si vijon:

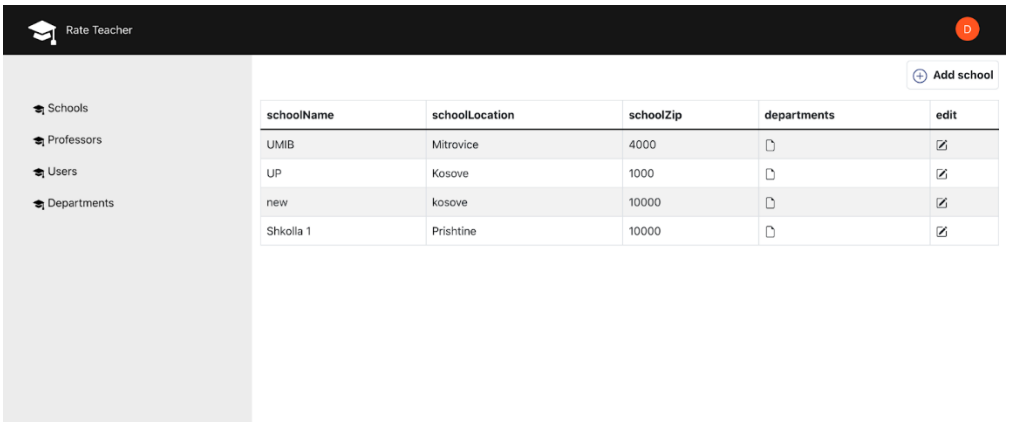

Figura 34. Kontroll paneli i aplikacionit

Sic shihet në figurën 25 në anën e majtë shihet lista e tabelave të databazës, perdoruesit me rolin admin përmes këtij paneli mundësohet editimi, shtimi dhe fshierja e të gjitha entiteteve të aplikacionit.

Një modal editimi në kontroll panel duket si vijon:

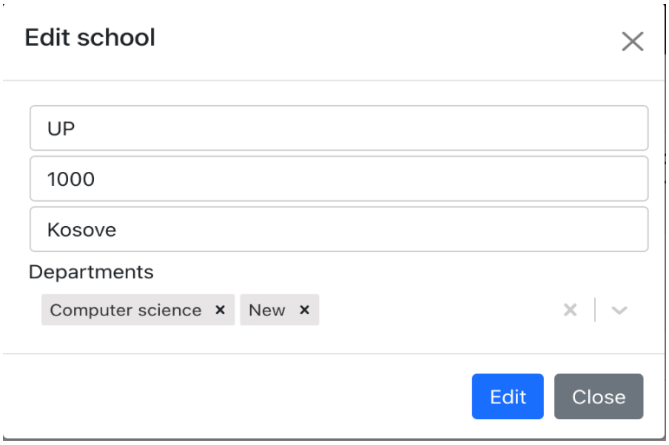

Figura 35. Modal editimi ne kontroll panel

4.4 Hooks e përdorura në zhvillimin e aplikacionit

Aplikacioni 'Rate Teacher' është zhvilluar duke përdorur komponentet e bazuara në funksion, dhe gjatë zhvillimit janë përdorur disa nga hooks që i cekëm në pjesën e parë të punimit. Në këtë pjesë të punimit do të demostrojë dhe sqaroj kodin se si kemi përdorur këto hooks.

- useEffect

  Ky hook është njëri nga hooks më të përdorur dhe është i paevitueshëm përdorimi i tij sa herë që përdorimim komponentet e bazuara në funksion. Ekzekutohet në tri raste: kur DOM-i renderohet në herën e parë, sa herë që ndonjë state pëson ndryshime dhe sa herë që deshirojmë ta largojmë një element nga DOM-i. Në vijim do të shohim se si kemi përdorur 'useEffect' në komponentën 'Rate.tsx'.

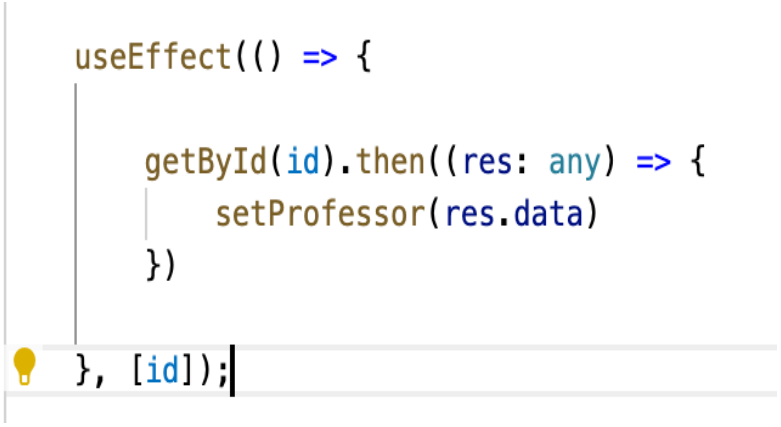

Figura 36. Perdorimi i useEffect

  useEffect pranon dy argumente, një funksion sic e shohim në figurën e mësipërme funksioni ku është thirrur funksioni tjetër getById më anë të së cilit marrim të dhënat e profesorit specifik dhe i ruajmë në gjendjen më emër 'professor'. Argumenti tjetër është një vektor i varësive, sa herë që njëri nga elementet në këtë vektor ndryshon thirret prapë funksioni brenda useEffect-tit, në rastin tonë sa herë që variabla 'id' ndryshon funksioni getById thirret prapë. Pra, useEffect-i i mësipërm thirret në dy raste: kur komponentetja renderohet dhe kur 'id' ndryshon.

- useState

Ky hook na mundëson të ndjekim gjendjen në një komponente funksion. Gjendja në përgjithësi i referohet të dhënave ose ‘properties’ që duhet të ndjeken në një aplikacion.

```
const [user, setUser] = useState<any>({userRoleName: 'user'})
return <>
    <div className={'mainWrapper'}>
        <div className={'wrapper'}>
            <div style={{flex: 1, textAlign: "center", color: 'white'}}>
                <div className={'leftSide'}>
                    <MortarboardFill height={100} width={100} color={'white'} onClick={() => navigate('/')}/>
                    {/*<p>Login to continue</p>*/}
                    <div style={{textAlign: 'left', width: '400px'}}>
                        <DefaultInput type='text' className='mb-3' placeholder={"Name"}
                                      onChange={(e) => setUser({...user, firstName: e.target.value})}/>
                        <DefaultInput type='text' className='mb-3' placeholder={"Surname"}
                                      onChange={(e) => setUser({...user, lastName: e.target.value})}/>
                        <DefaultInput type='text' placeholder='UserName' className='mb-3'
                                      onChange={(e) => setUser({...user, userName: e.target.value})}/>
                        <DefaultInput type='email' placeholder='E-mail' className='mb-3'
                                      onChange={(e) => setUser({...user, email: e.target.value})}/>
                        <DefaultInput type='password' placeholder='Password' className='mb-3'
                                      onChange={(e) => setUser({...user, password: e.target.value})}/>
                        <div style={{display: 'flex', justifyContent: 'space-around', marginTop: '20px'}}>
                            <MyButton title="Register" className="mb-4"
                                      style={{width: '10rem', backgroundColor: '#283779', border: 'none'}}
                                      onClick={() => props._register(user)}></MyButton>

                        </div>
                    </div>

                </div>
            </div>
```

Figura 37. Perdorimi i useState

useState merr një vlerë fillestare si argument në këtë rast një objekt me vlerën e userRoleName ‘user’ dhe kthen një grup me dy elementë: gjendjen aktuale në këtë rast user dhe një funksion për ta përditësuar atë, setUser. Ky shembull i përket ‘Register Template’, sic e shohim në kodin më sipër sa herë që një input i formës së regjistrimit plotësohet, ne përditësojmë gjendjen e userit me vlerën e plotësuar.

5. Hooks, e ardhmja e zhvillimit në React

Hooks ofrojnë një mënyrë për të shkruar një kod më të lexueshëm dhe të mirëmbajtur, dhe gjithashtu lejojnë ripërdorim më të mirë të kodit. Hooks tani konsiderohen si mënyra e rekomanduar për të shkruar komponentë në React dhe e ardhmja e zhvillimit të React mund të përfshijë një përdorim më të madh të Hooks.

Ndërsa Hooks po përdoren më gjerësisht, pritet që të zhvillohen më shumë biblioteka dhe mjete për t'i mbështetur ato. Trendi i Hooks është në rritje. Ato janë pranuar gjerësisht në komunitet dhe shumica e zhvilluesve po i përdorin ato në projektet e tyre. Edhe veçoritë e reja të React po zhvillohen duke përdorur hooks. Ka disa arsye pse Hooks konsiderohet të jetë e ardhmja e zhvillimit të React:

- Lexueshmëria dhe mirëmbajtja e përmirësuar:
  Hooks e bëjnë më të lehtë kuptimin e logjikës së një komponenti duke e ndarë atë në funksione të vogla dhe të ripërdorshme. Kjo përmirëson lexueshmërinë e kodit dhe e bën më të lehtë arsyetimin për sjelljen e komponentit.
- Ripërdorim më i mirë i kodit:
  Hooks ju lejojnë të ndani logjikën e gjendjes midis komponentëve, duke e bërë më të lehtë ripërdorimin e kodit në aplikacionin tuaj. Kjo mund të çojë në zhvillim më efikas dhe një bazë kodi më të qëndrueshme.
- Komponentët klasë janë më pak të preferuar:
  Komponentët e klasës mund të jenë të vështirë për t'u kuptuar, veçanërisht kur kemi të bëjmë me metoda komplekse të gjendjes dhe ciklit jetësor. Hooks ofrojnë një mënyrë më të pastër dhe më intuitive për të menaxhuar gjendjen dhe efektet anësore në komponentët funksionalë.
- Performancë më e mirë
  Hooks mund të çojnë në performancë më të mirë sepse ju lejojnë të ndani një komponent në funksione më të vogla dhe më të specializuara. Kjo mund të ndihmojë në reduktimin e numrit të ri-renderimeve dhe përmirësimin e performancës së përgjithshme të aplikacionit

- React Team dhe mbështetja e komunitetit:
  Vetë ekipi i React ka rekomanduar përdorimin e hooks dhe ata po punojnë në mënyrë aktive në përmirësimin e hooks dhe për t'i bërë ato më të qëndrueshme, gjë që është një tregues i qartë se ata shohin të ardhmen e zhvillimit të react në Hooks.

Pra, është mjaft e qartë se React Hooks janë e ardhmja e zhvillimit të React.

6. Përfundim

Në këtë punim, u njoftuam më shumë me React Hooks, benefitet dhe problemet që hooks zgjidhën, rregullat dhe praktikat më të mira të përdorimit të React Hooks, po ashtu përshkruajtëm aplikacionin 'Rate Teacher' i cili u zhvillua duke përdorur komponentet funksion dhe disa nga React Hooks, demostruam dhe shpjeguam se si janë përdorur këto hooks.

Pamë që React Hooks u japin komponentëve funksion akses në pjesën më të madhe të funksionalitetit të React që më parë disponohej vetëm me komponentët e klasës. Hooks gjithashtu u japin zhvilluesve një sintaksë më të thjeshtë për përdorimin e gjendjes, kryerjen e detyrave në përgjigje të eventeve të ciklit jetësor dhe ripërdorimin e kodit.

Si përfundim, ky punim synonte të eksploronte përfitimet dhe rastet e përdorimit të React Hooks. Nëpërmjet një analize të literaturës dhe demostrimit të shembujve të përdorimit në aplikacionin 'Rate Teacher', u tregua se Hooks ofrojnë një sërë avantazhesh ndaj komponentëve të klasës, duke përfshirë përmirësimin e lexueshmërisë dhe mirëmbajtjes, ripërdorimin më të mirë të kodit dhe performancën e përmirësuar. Për më tepër, mbështetja e ekipit React dhe e komunitetit ndaj Hooks tregojnë se Hooks janë e ardhmja e zhvillimit të React.

Ky punim tregon se Hooks janë një mjet i fuqishëm për ndërtimin e aplikacioneve të fuqishme dhe efikase në React dhe ia vlen të merren parasysh kur zhvillohen projekte të reja ose rifaktorohen ato ekzistuese.

Literatura

[1]. “The Best Guide to Know What Is React”, *Simplilearn.com* https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs [qasur me 01.12.2022]

[2]. “What is React.js? (Uses, Examples, & More)”, *HubSpot* https://blog.hubspot.com/website/react-js [qasur me 01.12.2022]

[3]. “React.js! Let’s Talk About It…”, *IWork Technologies* https://www.iworkplc.com/post/react-js-let-s-talk-about-it# [qasur me 05.12.2022]

[4]. “The benefits of ReactJS and reasons to choose it for your project”, *Peerbits* https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html [qasur me 08.12.2022]

[5]. “ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET”, *Microsoft* https://learn.microsoft.com/en-us/archive/msdn-magazine/2013/november/asp-net-single-page-applications-build-modern-responsive-web-apps-with-asp-net [qasur me 08.12.2022]

[6]. “SPA-Single Page Application”, *MDN Web Docs* https://developer.mozilla.org/en-US/docs/Glossary/SPA [qasur me 08.12.2022]

[7]. “ReactJS Components”, *GeeksForGeeks* https://www.geeksforgeeks.org/reactjs-components/ [qasur me 10.12.2022]

[8]. “Components and Props”, *ReactJS* https://reactjs.org/docs/components-and-props.html [qasur me 10.12.2022]

[9]. “ReactJS | State in React”, *GeeksForGeeks* https://www.geeksforgeeks.org/reactjs-state-react/ [qasur me 11.12.2022]

[10]. “Hooks at a Glance”, *ReactJS* https://reactjs.org/docs/hooks-overview.html [qasur me 12.12.2022]

[11]. “A deep dive into React Hooks”, *Coder Society* https://codersociety.com/blog/articles/react-hooks [qasur me 15.12.2022]

[12]. “Rules of React Hooks”, *CoderPad* https://coderpad.io/blog/development/rules-of-react-hooks/ [qasur me 16.12.2022]

[13]. “React Hooks best practices in 2022”, *BoscTechLabs* https://bosctechlabs.com/react-hooks-best-practices-2022/ [qasur me 18.12.2022]

[14]. “Best practices with React Hooks”, *SmashingMagazine* https://www.smashingmagazine.com/2020/04/react-hooks-best-practices/ [qasur me 18.12.2022]

[15]. “10 React Hooks Explained”, *Dev* https://dev.to/abhisheknaiidu/10-react-hooks-explained-3ino [qasur me 21.12.2022]

[16]. “All React Hooks and concepts in a single post”, *Dev* https://dev.to/nyctonio/all-react-hooks-and-concepts-in-a-single-post-1daf [qasur me 25.12.2022]

[17]. “React Hooks and its advantages”, *Board Infinity* https://www.boardinfinity.com/blog/react-hooks-and-its-advantages/ [qasur me 25.12.2022]

[18]. “React Hooks benefits and lessons learned” , *Confluent* https://www.confluent.io/blog/moving-to-react-benefits-and-lessons-learned-building-confluent-cloud-ui/ [qasur me 02.01.2023]

[19]. “What Is Nest.js? A Look at the Lightweight JavaScript Framework”, *Kinsta* https://kinsta.com/knowledgebase/nestjs/ [qasur me 05.01.2023]

[20]. “MongoDB tutorial”, *W3Schools* https://www.w3schools.com/mongodb/index.php [qasur me 07.01.2023]

Tabela e figurave