Developing an Expense Tracking Application using React and Node.js

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor's thesis

Degree Programme in Computer Applications
fall, 2022

Arpit khandelwal

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus / Computer Applications Tiivistelmä

Tekijä Arpit Khandelwal Vuosi 2022

Työn nimi **Kustannusseurantasovelluksen kehittäminen React ja Node.js:n avulla**

Ohjaaja Lasse Seppänen, Elina Hietaranta

TIIVISTELMÄ

Aiemmin LAMP-pino (Linux, Apache, MySQL, PHP tai Perl) ja Java-pohjaiset sovellukset hallitsivat Aiemmin LAMP-pino (Linux, Apache, MySQL, PHP tai Perl) ja Java-pohjaiset sovellukset hallitsivat verkkokehitystä (Java EE, Spring). Nämä pinot sisältävät useita ohjelmointikieliä, joita yksittäisen kehittäjän voi olla vaikea ymmärtää. Kehittäjä voi nyt harjoittaa sekä verkkosovelluksen etu- että taustatoimintoja verkkoteknologioiden viime vuosien aikana tapahtuneen edistyksen ansiosta.

Opinnäytetyön tarkoituksena oli analysoida verkkopohjaisen kuluseurantasovelluksen vaatimuksia. Figmaa hyödynnettiin web-sovelluksen prototyypin testaamiseen ja kehittämiseen. Käyttöliittymän käyttöliittymä luotiin Reactilla. Opinnäytetyö keskittyy siihen, miten verkkosovelluksen käytettävyyttä voidaan lisätä suuremman yleisön tavoittamiseksi. Opinnäytetyön teoreettisessa osassa käsiteltiin kustannusseurantaa, sen peruskomponentteja ja sen käyttöönottonopeutta. Seuraavassa on kuvaus kuluseurantaverkkosovelluksen edellytyksistä ja yleisistä ominaisuuksista. Opinnäytetyön käytännön osa sisältää web-sovelluksen Figma-designprototyypin sekä tarvittavan ympäristön ja riippuvuuksien asennuksen webapp-etu- ja back-end-prototyypeille Reactin ja Node.js:n avulla.

Opinnäytetyössä analysoitiin halutuimmat kuluseurantaverkkosovelluksen ominaisuusvaatimukset tekemällä kattava kirjallisuuskatsaus ja markkinatutkimus suosituimmasta kuluseurantaverkkosovelluksesta. Tulosten mukaan React on erinomainen verkkosovelluskehityskehys, mutta Figma on intuitiivinen UI/UX-prototyyppityökalu. Opinnäytetyön materiaali auttaa myös lukijoita ymmärtämään kustannusseurantasovelluksen ominaisuuksia ja saavutettavuusvaatimuksia sekä opastaa heitä läpivientiprosessissa.

Avainsanat Kustannusseuranta, React, Node.js, MongoDb.

Sivuja 37 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology Abstract

Author Arpit Khandelwal Year 2022

Subject Developing an Expense Tracking Application using React and Node.js

Supervisors Lasse Seppänen, Elina Hietaranta

ABSTRACT

In the past, the LAMP stack (Linux, Apache, MySQL, PHP, or Perl) and Java-based applications dominated web development (Java EE, Spring). These stacks contain multiple programming languages, which might be difficult for a single developer to comprehend. A developer can now engage in both the front-end and back-end operations of a web application due to advancements in web technologies over the past few years.

The purpose of the thesis was to analyze the requirements for a web-based expense tracking application. Figma was utilized to test and develop the prototype webapp. The front-end user interface was created using React. The thesis focuses on how to increase the accessibility of a webapp to reach a larger audience. In the theoretical portion of the thesis, the expense tracker, its basic components, and its rate of adoption were discussed. Following is a description of the expense tracker web application's prerequisites and general features. The practical element of the thesis includes the Figma design prototype of the webapp and the installation of the necessary environment and dependencies for the front-end and back-end webapp prototypes using React and Node.js.

The thesis study analyzed the most desired expense tracker webapp feature requirements by doing a comprehensive literature review and market research on the most popular expense tracker webapp. According to the findings, React is an outstanding webapp development framework, but Figma is an intuitive UI/UX prototyping tool. The material of the thesis will also assist readers in comprehending the features and accessibility requirements of the cost tracker app, as well as guide them through the implementation process.

Keywords Expense tracker, React, Node.js, MongoDb.

Pages 37 pages

Glossary

API	Application Programming Interface
Apps	Applications
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
JSX	JavaScript Syntax Extension
NPM	Node Package Manager
NPX	Node Package Execute
Rest	Representational State Transfer
UI	User Interface
URL	Uniform Resource Location
VS Code	Visual Studio Code

Contents

Figures

1 Introduction

The spending tracker program is often referred to as an expense manager and a money manager. It helps keep an accurate record of your cash receipts and is often the outcome of inadequate financial management skills. (Gravier, 2022)

People frequently overpay without recognizing it, with potentially fatal results. A daily spending planner could help you keep track of your daily expenses. Then, each day, you will clearly understand where your money is going.

Interface Designing and Application Development are seeing significant development in today's Software Development environment. In addition, the design must be capable of reversing the course of design rather than adhering to the original standard procedures. The other modes, web applications, are one of the locations where web developers may exercise their creativity and expect their solutions to be more effective and versatile. This paper will outline the most engaging and stylish path to app development.

This thesis aims to render react.js and node.js to non-technical readers while building a front-end user interface and back-end for a money tracker app using React and node.js. To appreciate the current market needs, adequate research will be undertaken at the beginning of the thesis by examining the most prominent cost trackers and websites. Then, to enhance the program,while functioning, Figma will be used to construct prototypes.
To ensure a practical execution of this study, the thesis will address the following two research topics:

1. How important is an expense tracker?

2. how does it enhance our decision making in context to daily/monthly expenses?

3. What are the Requirement/Tools for building an expense tracker?

2 Expense Tracker

A mobile application for cost tracking enables users to monitor and classify their expenses across many bank and investment accounts and credit cards. These applications also provide budgeting tools, credit monitoring, mileage tracking, and guidance on increasing user net worth. (HAAN, 2022)

The most recent smartphone applications have made it easy to monitor spending patterns, track costs, and manage money. For example, a monthly spending tracker app automates the recording of transactions, summarization of expenses by category, and monitoring progress toward targets. Several practical applications are available, but the finest tools are those actively used. (Khizhniakova, 2022)

2.1 Importance of Expense Tracker

The creation of expenditure monitoring apps is popular among organizations for tracking their spending and managing their budgets effectively. The following list summarizes some of the key elements used in the Expenses tracker.

2.1.1 Financial Conciousness

An analysis of a person's expenses, such as a week or a month, reveals a spending habit. It allows the user to understand better how someone spends their money. Additionally, it puts users in control, allowing them to change spending based on where they need to spend more money and where to cut costs. In addition, users can make any necessary modifications. (Buchenau, 2020)

2.1.2 Identifying Problem Areas

Users can better understand their finances as they monitor expenditures over time. That makes it much easy to identify areas of excessive expenditure. For example, using a credit card to make small purchases may seem like little expense when charged regularly. However, after the user

sums up how much the user spends on eating out, coffee, lottery tickets, or whatever the habit is, the Person may be surprised. (Caldwell, 2022)

2.1.3 Building A Better Budget

Many individuals attempt to create a budget without monitoring their expenditures, but this is a recipe for failure. In any case, it is doubtful that one will be able to adhere to the budget if users are grossly unrealistic about how much user will spend on entertainment, eating out, and another discretionary spending. On the other hand, if users have watched their expenditures and know where their money is going, they may make intelligent and sustainable cutbacks in desirable areas. (Buchenau, 2020b)

2.2 Influence Of Expense Tracking On Spending Decision

Managing a budget becomes increasingly important when the monthly revenue of businesspeople fluctuates, making budgeting more difficult. Secure spending tracking application helps collect receipts, maintain income records, create a budget, handle taxes, measure income vs. expenses, sync cards, and more. Developing apps for expense monitoring is currently one of the most popular trends, and many individuals are already utilizing them. In the United States, 45.5% of people have no retirement savings. As a result, they may have difficulties after retirement. (Duggal, 2021)

Figure 1 Graph of expense tracking adoption rate (Duggal, 2021)

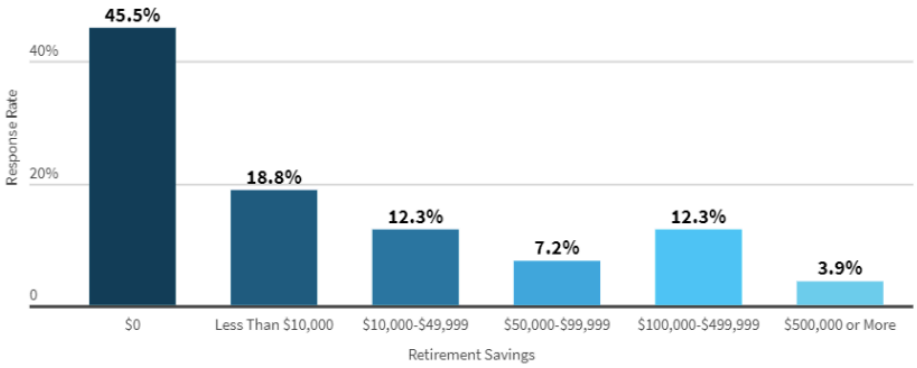


Figure 1 above, which consists of a poll reveals that 26% of Canadians scored below 50 on a financial well-being scale. 7 percent of this 26% have scores below 30 and are struggling significantly, while 19 percent have scores between 31 and 50 and are struggling slightly. (Duggal, 2021)

2.3 Functional Requirement For Building An Expense Tracker

The functional requirements for mobile apps specify what must be implemented in a system or product and how users must interact with the program. They determine the system's functionality.

The System must meet the specified requirements:

i. Must be able to register new users.
ii. User can manually update transactions inside the application.
iii. The User is able to inspect the transactions.
iv. The user is permitted to remove or modify any transaction.
v. The user can apply a filter for one-week and one-month transactions.

3 Software Requirements For Expense Tracking App

SRS is a document that outlines the nature of a program, application, or project. Simply said, an SRS document is a project's manual, provided it is produced before a project or application is initiated. (Bandakkanavar, 2022)

The following products require the following software:

1. Visual Studio Code
2. Language: React, Node.js
3. Figma.
4. Database used MongoDB database.
5. MongoDB Compass

3.1 Visual Studio Code Editor

Microsoft's Visual Studio is an Integrated Development Environment (IDE) for creating GUIs (Graphical User Interfaces), consoles, Web applications, online apps, mobile apps, cloud services, and web services, among other things. This IDE enables the creation of both managed code and native code. It utilizes Microsoft software development platforms such as Windows Store, Microsoft Silverlight, and Windows API, among others. It may be used to write code in C#, C++, VB (Visual Basic), Python, and JavaScript, among other languages. It supports thirty-six distinct programming languages. It is compatible with both Windows and macOS. (GeeksforGeeks, 2022)

3.2 React

An early prototype of React, created by a Facebook software engineer named Jordan Walke and dubbed FaxJS, was originally introduced in 2011. In 2012, Walke completed the prototype and invented React, which Facebook and Instagram quickly adopted that same year. React became available for Ruby on Rails and Python applications after being open-source in 2013. 2015 brings the release of React Native, an extension of React for mobile programming on Android and iOS. Since then, React has continually improved and added new user-facing capabilities with each new version throughout the years. (Hámori, 2022)

React is a well-known tool that can be applied on desktop and mobile platforms and has a variety of unique capabilities. One of them is the ability to design dynamic, rather than static, websites that may update and render data with each user input, enabling a seamless UI without needing to reload the entire page whenever a change is made. That is because React's structure is based on components, which enables the construction of complicated UIs. After all, component logic is written in JavaScript.

3.2.1 NPM/ Node Package Manager

NPM is an abbreviation for node package management, which is an online directory containing already registered open-source products. When installed with the NPM command npm install into an application, NPM modules consume the different functionalities as a third-party package.

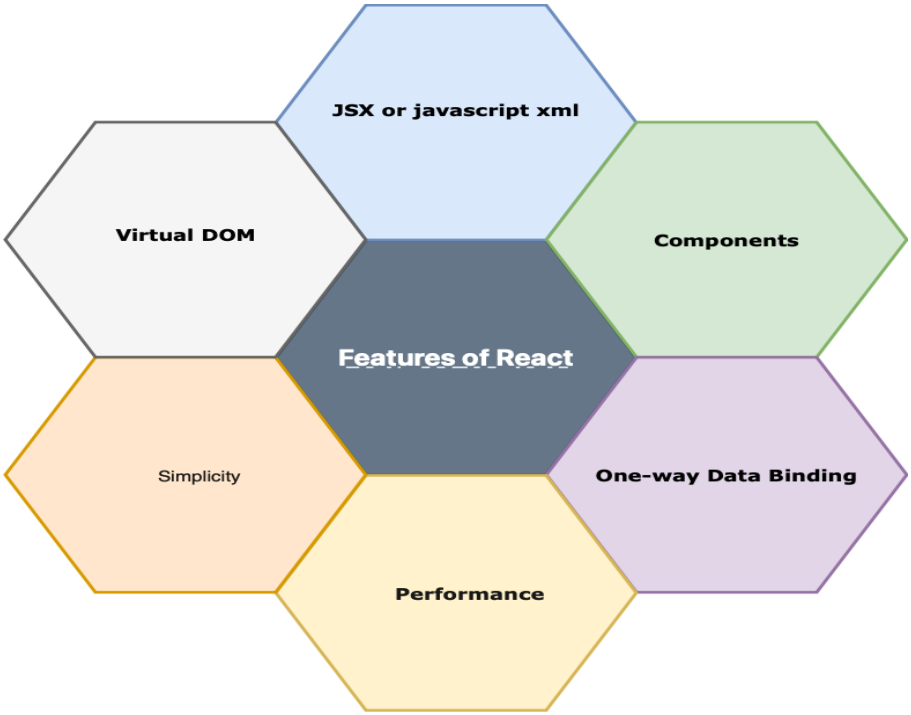
Third-party NPM modules, such as react-strap, material-UI, redux-form, and so on, are the unit of logic for a given functionality or a whole library. This post will teach you how to load NPM modules using the NPM command or GitHub source code. NPM is optimized and operated based on the following roles:

It may also help with package installation, uninstalling, version control, and server-driven management. Additionally, NPM is used to manage the dependencies required for projects to execute. (Singhal, 2020)

3.2.2 React Features

ReactJS is quickly becoming the most popular JavaScript framework among web developers. It is a vital component of the front-end ecology. (Savaram, 2022) ReactJS's key characteristics are as follows in Figure 2.

Figure 2 Picture of react features



Virtual DOM

The virtual DOM object is identical to the actual DOM object. The virtual DOM is comparable to unidirectional data binding. If any adjustments are made to a web application, the complete UI is re-rendered in a virtual DOM representation.

It evaluates the distinction between the new DOM and the existing DOM representation. Once the comparison is complete, the actual DOM will be updated to reflect the changes. The program will execute quickly and will not waste memory. (GeeksforGeeks, 2021)

JXS/ JavaScript XML

It is a syntax for markup that specifies the look of the application's user interface. It makes the syntax identical to HTML and helps developers construct React components.

JSX is one of the most excellent aspects of React JS since it makes it simple for developers to construct the building pieces. (GeeksforGeeks, 2021)

Program code 1 Explain Of JSX

```
const myElement = <h1>I Love JSX!</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

The Program code 1 React component displays to the DOM an h1 element containing the words "I Love JSX!" It employs JSX, a syntactic extension to JavaScript that enables us to create HTML-like code in JavaScript files.

Components

Components are fully independent and reusable programming fragments. They perform the same purpose as JavaScript functions but return HTML and operate independently. Class components and Function components are the two sorts of components. For example, the components of ReactJs is as follows.

- Function Component

Class components also return HTML and behave similarly to Function components. However, Function components may be written with far less code and are simpler to understand.

Program code 2 For Functional Component

```
function Car() {
  return <h2>Hi, I am a Car!</h2>;
}
```

- Class Component

A component class must have the term extends React Component. This line defines an inheritance from React. Component and offers access to its functions to use components. Additionally, the component requires the HTML-returning render() method.

Program code 3 For Class Component

```
class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}
```

One-Way Data Binding

Data flow is unidirectional in ReactJS, often known as "one-way data binding." The advantages of one-way data binding provide the programmer with greater control over the entire program. However, additional capabilities are required if data travels in the opposite direction. Because components are intended to be immutable, and their data cannot be modified, this is the situation. The Flux architecture allows the unidirectional flow of data. As a result, the application becomes more adaptable, resulting in increased productivity. (Kai, 2022c)

Performance

This feature makes it superior to other frameworks currently available. That is because it controls a virtual DOM. The Document Object Model (DOM) is a programming interface that works with HTML, XML, and XHTML. The DOM is purely memory-based. As a result, when we construct a component, we do not immediately write to the DOM. Instead, we are developing virtual components that will become DOM elements, resulting in a more fluid and efficient user experience. (*ReactJS Features - Javatpoint*, n.d.)

Simplicity

ReactJS utilizes JSX files, which makes the application easy to develop and comprehend. We know ReactJS is a component-based solution that makes code easily reusable. That makes it easy to use and understand.

3.3 Node.js

Node.js is a server-side JavaScript runtime environment that executes JavaScript code. That enables programmers to create scalable network applications, as it can manage a large number of concurrent connections with high throughput. Node.js, for instance, generates an application that requires a constant connection between the browser and the server. (*About*, n.d.)

Node.js is frequently the primary option of software developers because it offers asynchronous and event-driven code execution, a single-threaded paradigm, and no buffering. JavaScript and RESTAPIs are used to construct applications in Node.js, combining the front-end and back-end codebases under a single programming language. However, For CPU-intensive applications, Node.js is not recommended.

3.4 Basic Of Express

Express is a flexible web application framework for Node.js. Compared to Node.js alone, Express.js includes extra capabilities that make web application development faster and simpler. It offers an extensive range of capabilities for both web and mobile apps. With a variety of HTTP utility methods and middleware, it is simple to develop robust APIs.

Program code 4 Express App example which returns "Hello world"

app.js

```
const express = require('express');
const app = express();

app.get('/', function(req, res) {
  res.send('Hello World!');
});

app.listen(8000, function() {
  console.log('Example app listening on port 8000!');
});
```

3.5 MongoDB

MongoDB is a document-oriented, open-source database built to efficiently store and process vast volumes of data. MongoDB is a NoSQL (Not just SQL) database because data is stored and accessed without using tables. (Botelho & Vaughan, 2020)

he MongoDB database was initially published in February 2009 and is created and managed by MongoDB.com under the SSPL (Server-Side Public License). It also offers official driver support for the most common programming languages, including C, C++, C#, Etc. Net, Go, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala, Swift, and Mongodb are examples of programming languages. For programmers to be able to construct an application utilizing any of these languages. Numerous organizations, like Facebook, Nokia, eBay, Adobe, Google, Etc., utilize MongoDB to store massive amounts of data. (GeeksforGeeks, 2021a)

3.6 Figma

Figma is a web-based interface design and graphics editing application. It may be used for graphic design, including wireframing websites, building mobile app interfaces, prototyping designs, and composing social media posts. Figma is unlike other graphic editing applications precisely because it operates directly in the user browser, which allows the user to access projects and begin designing from any computer or platform without the need to purchase additional licenses or install the software. (Perera, 2020)

Figma was used to design the Expenses tracker application, including the final prototype. It features several layout possibilities and an intuitive user interface. Thus no prior expertise was required throughout the creation process of this thesis.

4 Accessibility Characteristics Of The APP

Depending on the app's functionality, it is crucial to comprehend how all users may interact with the mobile application. Therefore, the application should include interaction support so users can interact as planned. They are using voice recognition, which may be the most effective way to ensure that blind users enjoy the app (Initiative, n.d.). Utilizing React JS Accessibility while developing an application or website is one of the most prevalent ways taken by businesses. That is due to the capabilities and adaptability that React JS provides. Now, there is no one-stop solution for making a website or an app accessible. All or most disabilities require attention due to their diversity.

The Internet is becoming an increasingly vital resource in various fields, including education, employment, government, business, healthcare, and recreation. The Web must be accessible to give people with varying abilities equal access and opportunities. The United Nations Convention on the Rights of Persons with Disabilities recognizes access to information and communication technologies, such as the Internet, as a fundamental human right (UN CRPD).

The European Accessibility Act is a new directive that creates a single set of accessibility standards for software, digital services, and hardware sold or used throughout the European Union. In doing so, the act intends to eliminate barriers caused by varied accessibility standards in EU member states and Enhance the internal market for items and services that are accessible. (Applause App Quality, Inc, 2022)

Accessibility promotes social inclusion for those with impairments and others, including elderly persons and individuals in rural places and emerging nations.

5 Design and Application Development

User interface the design of an application includes its colors, typefaces, and overall appearance. The user experience, in contrast, will always be centered on accurate functioning and usefulness. Application design is also one of the critical determinants of which mobile and online applications users will use regularly and which they will avoid in the future. (The Design Project & Flinn, n.d.)

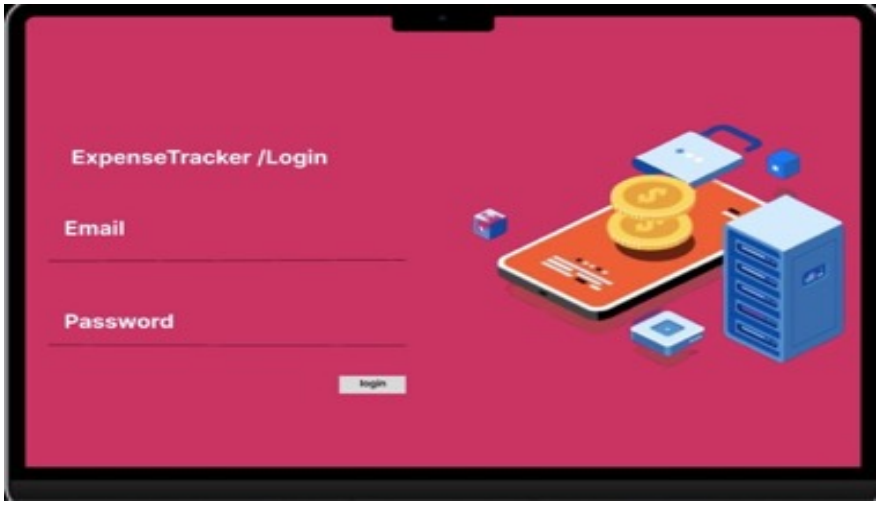
The installation of the Expenses tracker application will fall into three distinct phases: back-end development, front-end development. Due to the restricted scope of the thesis, it should only cover a few aspects of the project in depth. However, it can illustrate all the essential processes for a successful application implementation.

5.1 Figma Prototype For Expense Tracker App

Before beginning to code, the concept was created and designed in Figma to have a clear picture of how the Expense Tracker application should seem. Users can acquire access to an application by providing their username and password on the login screen.

5.1.1 Login Screen

Figure 3 Prototype of Login Screen



5.1.2 Registration page

The prototype of Registration page allows users to register for an application.

Figure 4 Prototype of Resigtration Page

5.1.3 Home Page

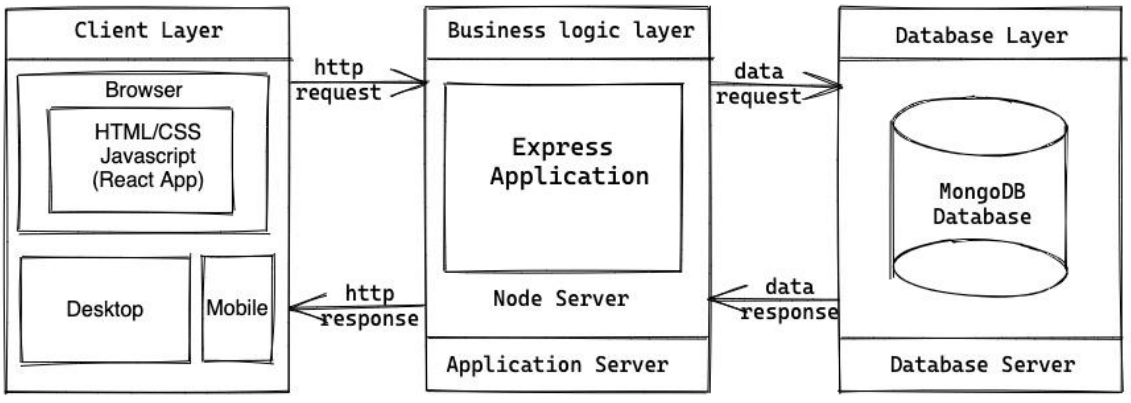
The home page will have a navigation bar at the top with connections to Analyse, including the User name and logout link, and show transaction records for users. Clicking the graph button will reveal Analyse transfer data. The pick time range and select type drop-down menus on the page on the data table simplify navigation by allowing the user to navigate between categories without difficulty. Moreover, a home page layout with Add button for the user's Transaction type, date, and reference information. For input fields, client-side form validation settings are intended. Consequently, random blank contributions are not permitted.

Figure 5 Prototype design of Home page

5.2 Architecture Of App

A Figure 6 below description of the patterns and methodologies utilized in application design and development. When creating an application, the architecture provides a road map and best practices to adhere to.

Figure 6 Architecture roadmap of app



Client view

These are the views (webpages) that are displayed to the client. The client layer will be developed with React, which uses a combination of Javascript, HTML, and CSS.

On this layer, end users attempt to access our application's features.

Application layer

This is the business logic layer, which is an Express application built atop Node.js. This application server acts as the link between the client and database layers. This layer accepts requests from clients and serves them by retrieving relevant data from the database layer.

Database layer

The mongoDB server is managed by the database layer. This mongoDB database contains all application data.

In a full-stack application, the user-accessible client layer (React application) requests web pages. The application server, which is the logical layer, receives these requests. It determines what data/web pages a client requires and sends a request to the database (database layer) to retrieve this data. Finally, the backend server (application layer) sends required data/web pages to the React application, which renders the data on the client's web page.

5.3 Front-End Development

Front-end development is a type of computer programming that focuses on coding and building visible website elements and functionalities. It involves ensuring that the aesthetic elements of a website are functioning. The front end is sometimes known as the "client-side" of an application. (Griffith, 2022b)

5.3.1 Expense Tracker Application Configuration

To create a react project, open the terminal in the code editor or the computer's command line. The most recent version of NPM is required to ensure that the most recent version of the create-react-app project command is utilized. After performing command 1 in the terminal and installing all necessary dependencies, a new Expense tracker-app project will be created. After changing directories, it is essential to browse to the main folder using command 2.

Command 1 Naming the react project and installing all required packages from terminal

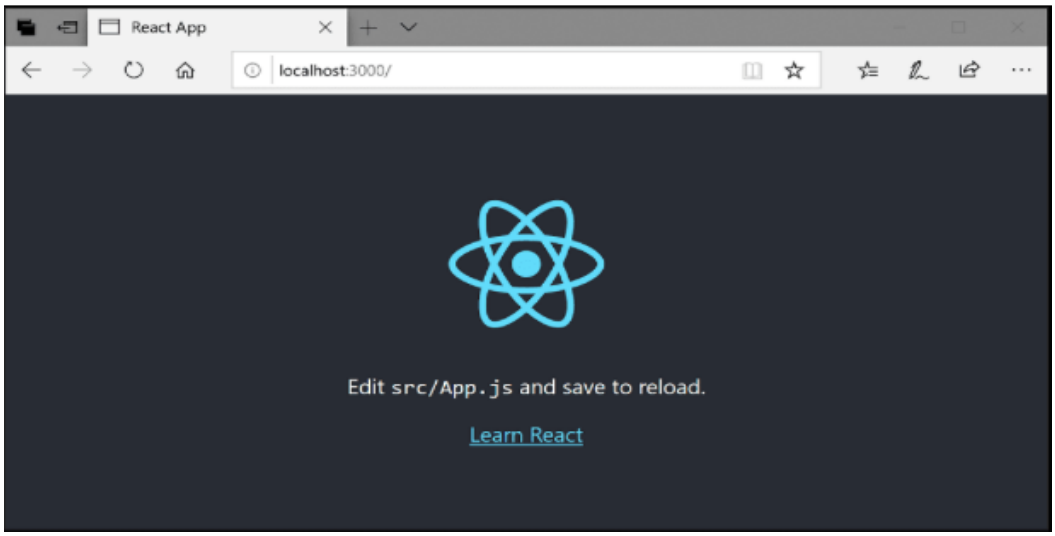
```
npx create-react-app expensetracker
```

Command 2 swap directories and browse to expense tracker app's main folder

```
Cd expencetracker
```

After generating a React.js application successfully, the command "npm start" ran to show the application in the user's default browser. The image below depicts how React.js appears on a web browser.

Figure 7 Example of stratup react app screen



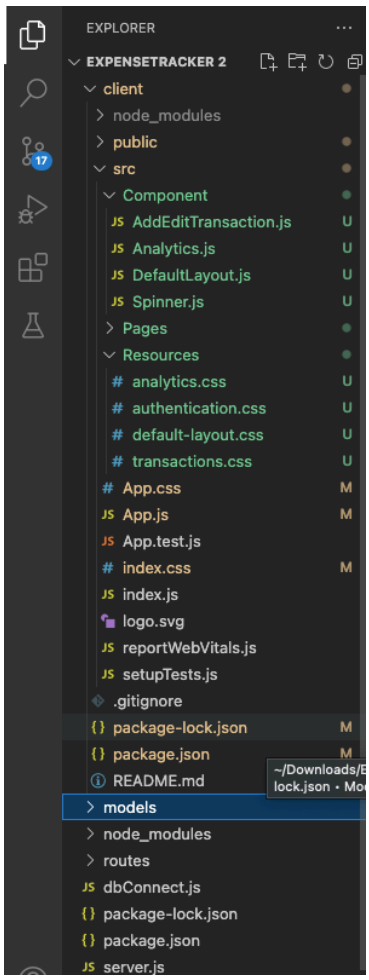
The index.js file in the src folder is the entry point for the whole program. The diagram below illustrates how a React.js application operates. First, the index.js file imports all required libraries, and then it renders the App.js root component using the ReactDom.render() function.

5.3.2 Directory Structures

As illustrated in

Figure 8 File structure of Expense Tracker app, the front-end development folders and files were included in the project structure. They are layered within one another to give the project a correct structure and make it easier to understand.

Figure 8 File structure of Expense Tracker app



The folder and file contents are detailed in detail below;

- The "public" folder contains an index.html page, a manifest.json file, and a logo image used for application deployment.

- The "components" folder contains an application's react components. These components are the building pieces of a program that describe the appearance of the User Interface.

- The "Resources" folder contains all necessary.css files for styling react component styles.

- A "App.js" file includes the application's base component. React treats all components hierarchically, with App /> being the highest-level component.

- An "Index.js" file is the entry point for all node apps describing what and where to render.

- The "models" folder is used to define the structure of our database and create a model for us to use when interacting with the database.

- The "package-lock.json" file enshrines the presently installed versions of a package; npm will utilize these precise versions when calling "npm install."

- The "package.json" file contains information about the project's metadata and dependencies.

- The "server.js" file is responsible for starting the server and connecting it to the database.

5.3.3 Configuring route in React app

SPA (Single Page Application) is currently a trend in the development of web applications due to its numerous exceptional characteristics. For example, depending on the component rendering situation, UIs may be rendered on a single page when an application employs the SPA technique. Furthermore, users can also determine which components appear by using the URL. React Router is the React.js library created for this purpose.

React Router is a framework for conventional URL navigation in React.js that enables UI and URL synchronization. It was created with a simple API, which expedites the resolution of URL-related issues. The installation instructions for the React Router library are shown in command 3.

Command 3 for installing Router-dom

```
npm install react-router-dom
```

Program code 5 to import react router

```
import {BrowserRouter , Navigate, Route , Routes} from 'react-router-dom';
```

The Program code 5 imports the BrowserRouter, Navigate, Route, and Routes components from the react-router-dom library. These components are used to create routes in a React application.

Program code 6 Example of react- router path

```
<BrowserRouter>
  <Routes>
    <Route path='/' element={<ProtectedRoutes><Home/></ProtectedRoutes>}></Route>
    <Route path='/login' element={<Login/>}></Route>
    <Route path='/register' element={<Register/>}></Route>

  </Routes>
```

Program code 6 is a react-router that lets the user move between pages. It also provides a secured route that verifies the user's login status before granting access to specific pages.

5.3.4 React Hooks

React Hooks are sophisticated function components that "hook into" React features. By default, a function component does not include a state; however, a React hook named useState may be used to keep the state during the component's lifetime. The useState hook and all other hooks may be utilized many times inside the same component. The Program code 7 shows a function component utilizing the useState hook.

Program code 7 Example of useState hook

```
function Home() {
  const [showEditAddTransactionModal,setshowEditAddTransactionModal]=useState(false);

  const [loading,setLoading]=useState(false);
  const [transactionsData,setTransactionsData]=useState([]);
  const [frequency,setFrequency]=useState('7');
  const [type,setType]=useState('all');
  const [selectedRange,setSelectedRange]=useState([]);
  const [selectedItemForEdit,setSelectedItemForEdit]=useState(null);
  const [viewType,setViewType]=useState('table');
  const { RangePicker } = DatePicker;
  const getTransaction=async()=>{
    try{
```

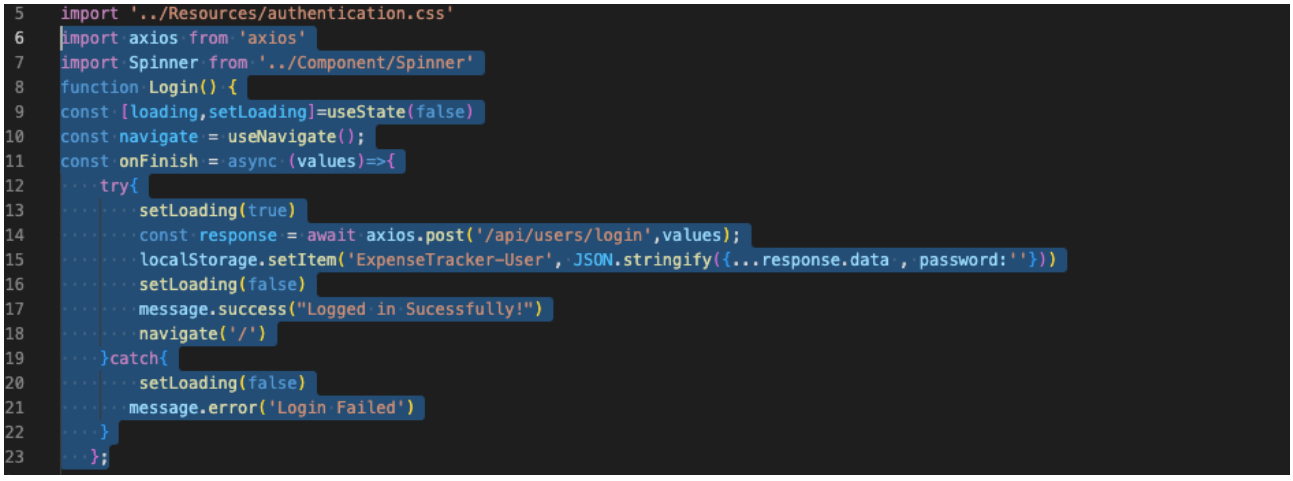
The useEffect hook may be used to construct life-cycle methods for function components that lack such methods by default. By default, the useEffect hook will execute a function for each component re-render; however, it may be altered to execute just for specific alterations. Program code 8 is an example of the useEffect hook.

Program code 8 Example of useEffect hook

5.3.5 React Axios

ReactJS's Axios library generates HTTP requests for external resources. On occasion, React applications may need to retrieve data from an external source. Such information is difficult to retrieve, so it may be displayed routinely on a webapp. Consequently, it facilitates the retrieval of the data, adding it to the state so that it can assist the application whenever it is required.

Program code 9 of Axios



Above Program code 9 of Axios **Error! Reference source not found.** refer This is the function component responsible for rendering the login page. It features a form with two fields: email and

password. When the user accepts the form, an Axis authentication request is sent to the backend server. If successful, it stores data in local storage and sends the user to the homepage. If the operation fails, it displays an error message.

5.3.6 Ant Design

For the Expense Tracker application, Ant Design React components will be utilized.
Ant Design is a design system that enhances the user experience of corporate applications by incorporating the ideals of nature and determinism. Using Ant Design, users may obtain access to a vast selection of great components for rapid application development, as well as samples and documentation.

Program code 10 of Ant Framework

Program code 10 is an example of a javascript import statement. It imports the message, Select, and Table antd library components. These are React components that can be used to render UI elements in our code.

5.4 Login Page

Login page is a web page where users can enter their credentials to access the application.
The Program code 11 is a try catch block that sends the user's email and password to the server. If it succeeds, it will save the user data in local storage and navigate to the home page. If it fails, it will display an error message.

Program code 11 of try-catch statement for login page

```
try{
    setLoading(true)
    const response = await axios.post('/api/users/login',values);
    localStorage.setItem('ExpenseTracker-User', JSON.stringify({...response.data , password:''}))
    setLoading(false)
    message.success("Logged in Sucessfully!")
    navigate('/')
}catch{
    setLoading(false)
    message.error('Login Failed')
  }
};
```

The

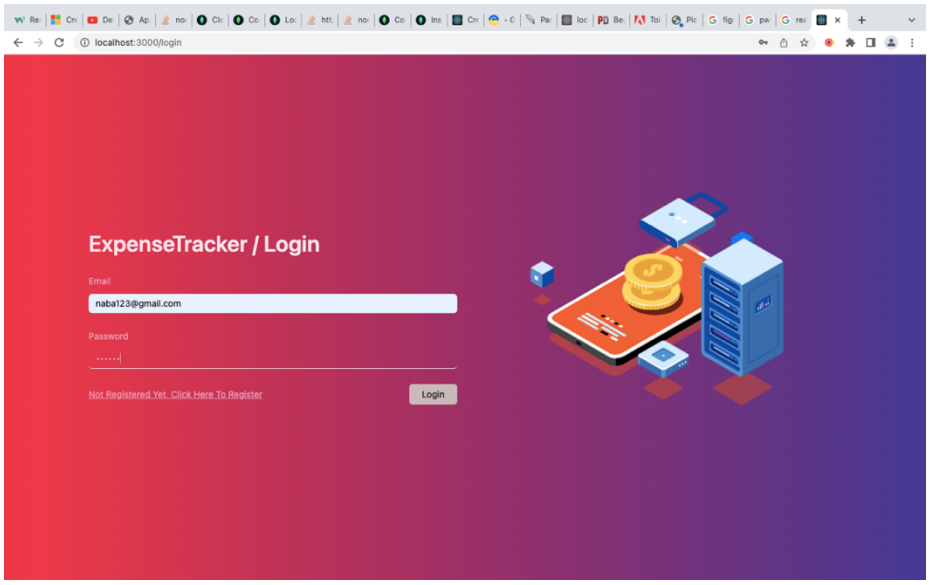
Program code 12

Program code 12 of Login form It uses the useState hook to store a boolean value that indicates if it's loading or not. It also uses the useEffect hook to check if there is already data in local storage, and redirects to home page if so. The component renders two columns with Bootstrap classes: one for login form, and another for an animation using Lottie player. When user clicks on submit button, it calls the function passed as prop (onFinish) which sends HTTP request to server with email and password entered by user. If successful, it saves some data in local storage then redirects to home page; otherwise shows error message using Antd library components (Form & Message).

Program code 12 of Login form

```
return (
  <div className='register'>
        {loading && <Spinner/>}
      <div className='row justify-content-center align-items-center w-100 h-100'>

          <div className='col-md-5'>
              <Form layout='vertical' onFinish={onFinish}>
              <h1>ExpenseTracker / Login</h1>


                  <Form.Item name='email' label='Email'>
                      <Input/>
                  </Form.Item>
                  <Form.Item name='password' label='Password'>
                      <Input type='password'/>
                  </Form.Item>

                  <div className='d-flex justify-content-between align-items-center'>
                      <Link to='/register'>Not Registered Yet, Click Here To Register</Link>
                      <button className='secondary' type='submit'>Login</button>
                  </div>

              </Form>
          </div>
          <div className='col-md-5'>
              <div className='lottie'>
                  <lottie-player src="https://assets3.lottiefiles.com/packages/lf20_ml0yft0o.json"  background="transparent"  sp
              </div>
          </div>

      </div>
  </div>
)
```

Figure 9 Screenshot of Login page display on browser



5.5 Resigtration Page

A signup page (also known as a registration page) allows people to register and receive access to the Expense Tracker app independently.

The Program code 13 below is the component responsible for rendering the registration page. It has a form that collects user information and transmits it to the server for registration. When the form is submitted, the onFinish method is called which then contacts an API endpoint to register the user.

Program code 13 of functional component for Signup page

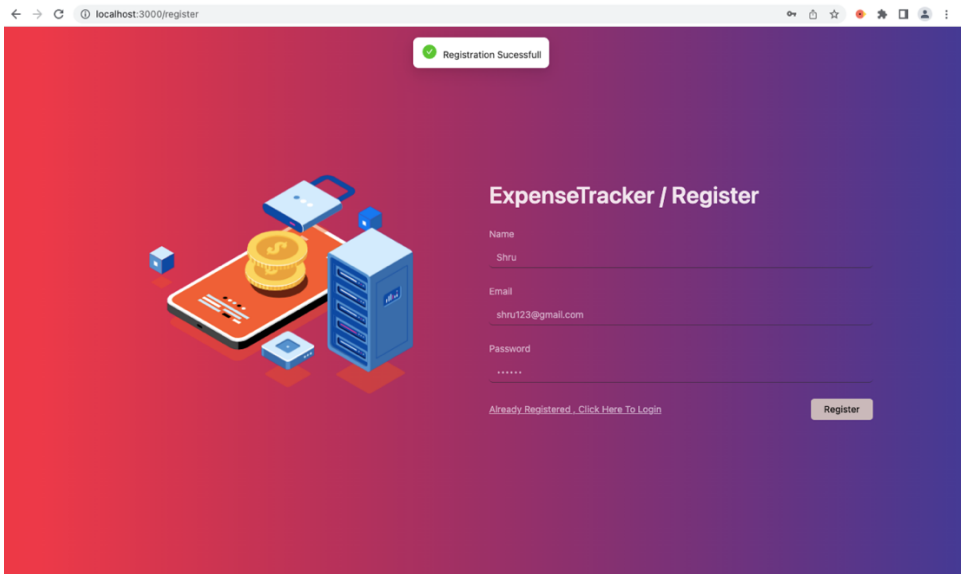
```
function Register () {
const [loading,setLoading]=useState(false)
const navigate = useNavigate();
const onFinish = async (values)=>{;
 try{
    setLoading(true)
    await axios.post('/api/users/register',values);
      message.success('Registration Sucessfull')
      setLoading(false)
 }catch(error){
    message.error('Something went wrong',error)
    setLoading(false)
 }
};
```

Program code 14 form of singup page

```
Register.js U
client > src > Pages > Register.js > Register
31      <div className='register'>
32          {loading && <Spinner/>}
33          <div className='row justify-content-center align-items-center w-100 h-100'>
34              <div className='col-md-5'>
35                  <div className='lottie'>
36                      <lottie-player src="https://assets3.lottiefiles.com/packages/lf20_ml0yft0o.json"  background="transparent"  speed="1"  loop  autoplay></lottie-player>
37                  </div>
38              </div>
39              <div className='col-md-5'>
40                  <Form layout='vertical' onFinish={onFinish}>
41                  <h1>ExpenseTracker / Register</h1>
42
43                      <Form.Item name='name' label='Name'>
44                          <Input/>
45                      </Form.Item>
46                      <Form.Item name='email' label='Email'>
47                          <Input/>
48                      </Form.Item>
49                      <Form.Item name='password' label='Password'>
50                          <Input type='password'/>
51                      </Form.Item>
52
53                      <div className='d-flex justify-content-between align-items-center'>
54                          <Link to='/login'>Already Registered , Click Here To Login</Link>
55                          <button className='secondary' type='submit'>Register</button>
56                      </div>
57
58                  </Form>
59              </div>
60
61          </div>
62      </div>
63      )
64  }
65
66  export default Register
```

As illustrated in the Program code 14 below It contains a form with three fields (name, email and password) and two buttons (one for registering and one for redirecting to the login page). The user can register by filling in the fields and clicking on the register button. If user is already registered, user can click on the second button to be redirected to the login page.

Figure 10 Screenshot of Signup button



A signup page allows people to register and receive access to the Expense Tracker app independently. In Figure 10 It contains a form with three fields (name, email and password) and two buttons (one for registering and one for redirecting to the login page). In addition, it provides a green popup for successful registration.

5.6 Home Page

As illustrate in the Program code 15 is a function that returns the home page of the application. It has state variables that are used to store data and functions that are used to manipulate data. The state variables are: showEditAddTransactionModal, loading, transactionsData, frequency, type, selectedRange, selectedItemForEdit and viewType. The functions are: getTransaction(), deleteTransaction() and useEffect().

Program code 15 Example of Function in home page

```
function Home() {
  const [showEditAddTransactionModal,setshowEditAddTransactionModal]=useState(false);

  const [loading,setLoading]=useState(false);
  const [transactionsData,setTransactionsData]=useState([]);
  const [frequency,setFrequency]=useState('7');
  const [type,setType]=useState('all');
  const [selectedRange,setSelectedRange]=useState([]);
  const [selectedItemForEdit,setSelectedItemForEdit]=useState(null);
  const [viewType,setViewType]=useState('table');
  const { RangePicker } = DatePicker;
  const getTransaction=async()=>{
    try{
```

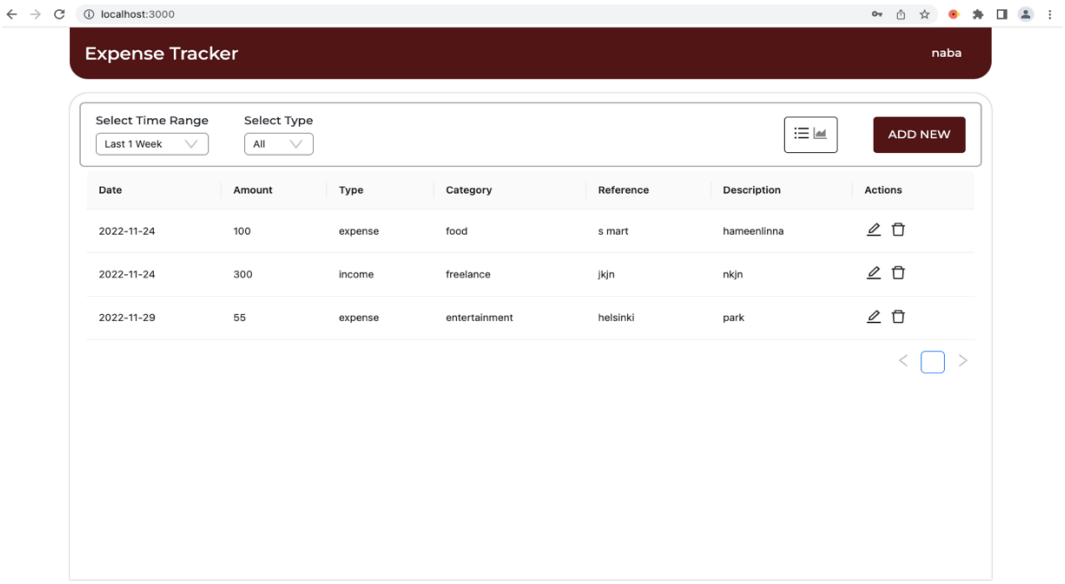
Program code 16 of try catch statement in home page

```
try{
    const user = JSON.parse(localStorage.getItem('ExpenseTracker-User'));
    setLoading(true);
    const response = await axios.post('/api/transactions/get-all-transactions',{userid : user._id,frequency,...(frequency==='custom' && {selectedRange}),type});
    setTransactionsData(response.data);
    setLoading(false);

}catch{
    setLoading(false);
    message.error('Something went wrong');
}
}
const deleteTransaction=async(record)=>{
    try{

    setLoading(true);
    await axios.post('/api/transactions/delete-transaction',{transactionId : record._id});
    message.success('Transaction Deleted Sucessfully');
    getTransaction();
    setLoading(false);

}catch{
    setLoading(false);
    message.error('Something went wrong');
}
}
useEffect(() => {
  getTransaction()
}, [frequency,selectedRange,type])
```

The purpose of the Program code 16 is to get all the transactions from the database and display them to the user. It also has a delete button which deletes a transaction when clicked. The useEffect hook runs this function whenever there is a change in frequency, selectedRange or type

The Figure 11 represents the home screen of the ExpenseTracker application, which contains all the transaction information supplied by users. It is the only page where registered users may use the application, including sign-up and sign-in capabilities. Typically, Analyze pages and the Add New Expenses button will feature a navigation bar at the top of the Data table and two selection fields for choosing the period and kind of transaction data to display. In the upper right-hand corner, the user's name will be shown; to log out, click the user's name.

Figure 11 Screenshot of home page



5.7 Analiytic Page

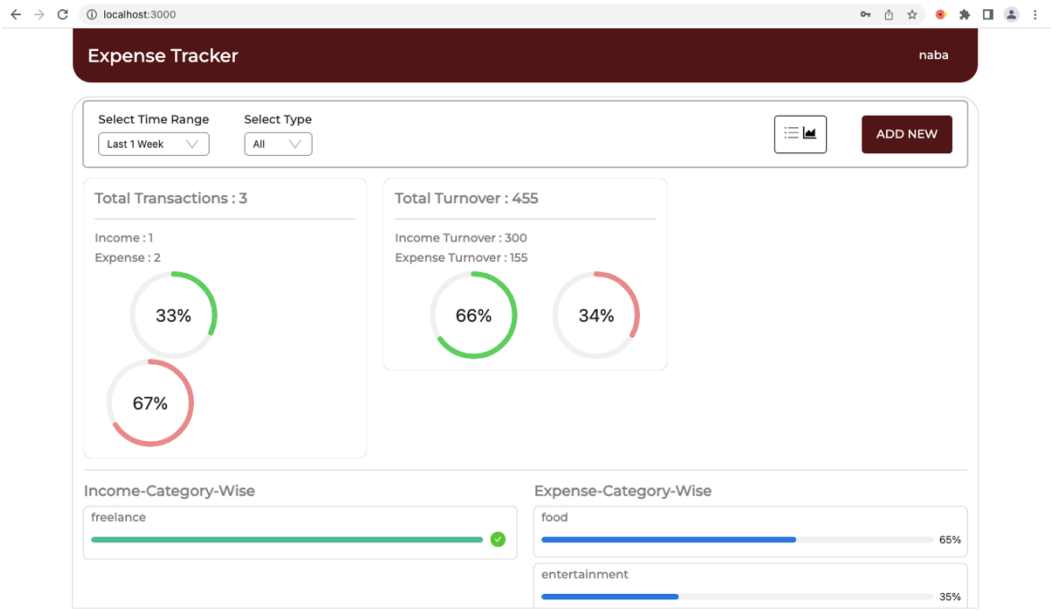
The Program code 17 bwlow is a function that takes in an array of objects as a parameter and returns the total number of transactions, the total income transactions, the total expense transactions, the percentage of income transactions to all transactions, the percentage of expense transactions to all transactions, the total turnover (total amount), the total income turnover (total amount for incomes), the total expense turnover (total amount for expenses), and finally it calculates and returns the percentages of both turnovers.

Program code 17 of Analiytics Function

```
function Analytics({transactions}) {

    const totalTransactions = transactions.length;
    const totalIncomeTransactions = transactions.filter(transactions => transactions.type==='income');
    const totalExpenseTransactions = transactions.filter(transactions => transactions.type==='expense');
    const incomeTrasactionPercent=(totalIncomeTransactions.length/totalTransactions)*100;
    const expenseTrasactionPercent=(totalExpenseTransactions.length/totalTransactions)*100;

    const totalTurnover = transactions.reduce((acc,transaction)=>acc + transaction.amount,0);

    const totalIncomeTurnover =  transactions.filter(transaction=> transaction.type==='income').reduce((acc,transaction)=>acc + transaction.amount,0);

    const totalExpenseTurnover =  transactions.filter(transaction=> transaction.type==='expense').reduce((acc,transaction)=>acc + transaction.amount,0);
    const incomeTurnoverPercent =(totalIncomeTurnover/totalTurnover)*100 ;
    const expenseTurnoverPercent =(totalExpenseTurnover/totalTurnover)*100 ;
```

In Figure 12 the analytics page above, a custom time range is available to choose from the time dropdown menu including the type of transactions. Total transactions and total turnover are calculated on the basis of income and expenses and presented on the graph on percentage basis.

Figure 12 Display the Analiytics page



5.8 Back-End Development

The Express.js library built on top of Node.js is used to construct the server for the application's back end. Following the configuration of MongoDB, the Mongoose library could store JSON data and connect it to MongoDB. Finally, routing-related logic and the front-end interaction endpoints appear.

5.8.1 Server Development

Primarily, a server is required to build JavaScript code on the backend.

Command 4 to initialize a project.

```
npm init
```

The above command 4 is used to create a new Node.js project. After answering basic questions about the project's description and the user's details, a package.json file is generated. Instead of utilizing the core Node.js HTTP core modules to create the server after project inception, the Express.js framework is utilized to leverage existing npm packages.

Command 5 Install express

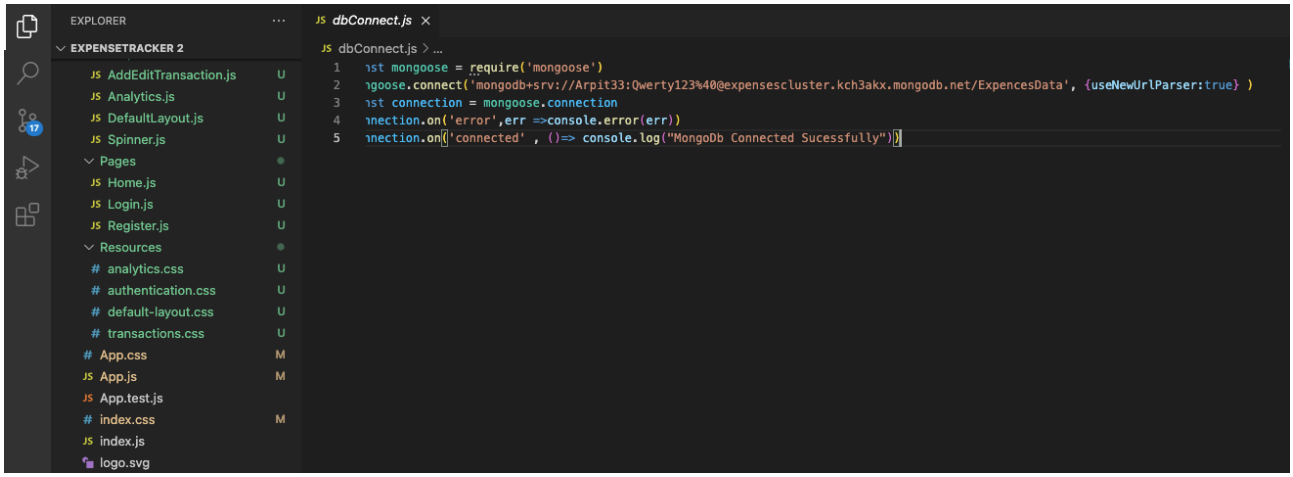
```
npm install express
```

After that, the app.js entry file gets built. This file contains all node packages and middleware.

5.8.2 Connecting to mongoDb

As illustrate Program code 18 below is a code snippet that connects to MongoDB Atlas. It uses the mongoose library to connect to the database. The connection string is passed as an argument in the connect function. The connection string is a URL that contains the username and password of your MongoDB Atlas account. It also includes the name of the cluster you created in MongoDB Atlas. We can connect to a database using mongoose by calling the connect () method. The first parameter is the connection string, and second parameter is an object with options for connecting to MongoDB.

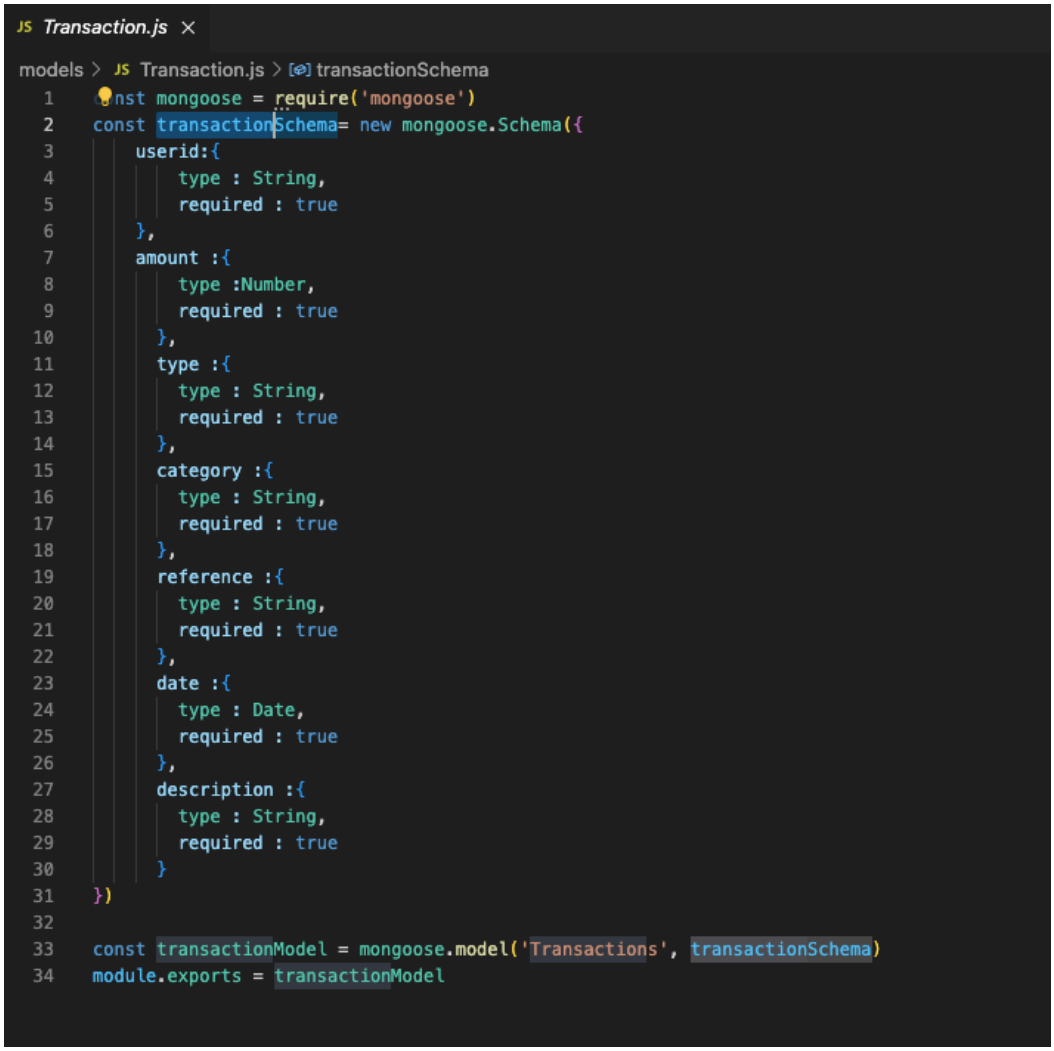
Program code 18 connecting with mongodb



5.8.3 Database Development

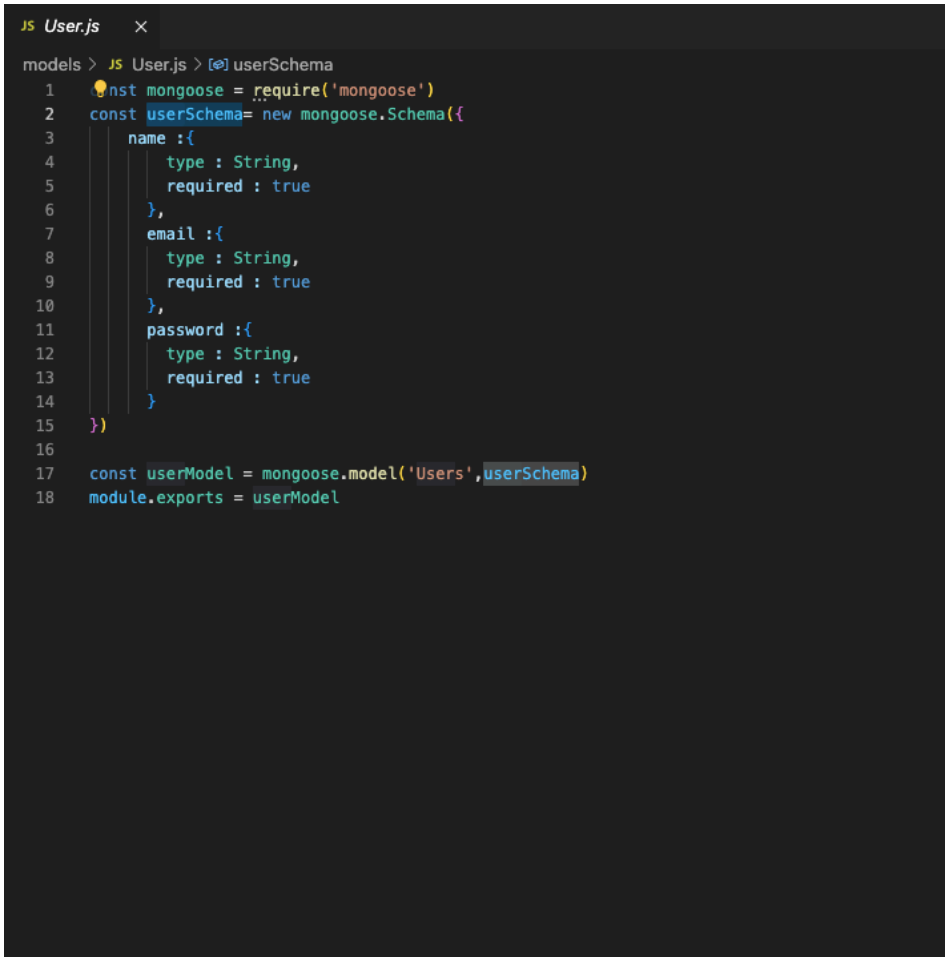
Installing the Mongoose node package enables the Express.js application to communicate with the MongoDB database. Mongoose is an Object Data Modelling tool that assists MongoDB with schema creation. Based on this interface, models are defined. The Schema enables users to specify the fields saved in each document and their validation methods and default values. In essence, the Schema is a blueprint for building the database. The application contains two schemas: user and transaction.

Program code 19 Database Transaction Schema



The Program code 19Program code 19 below to displays the structure of the transaction schema in detail. The schema has various fields related to the transaction, including user ID, amount, type, category, date, references, Etc. In addition, it illustrates the relationship between transaction and User models in the database. For instance, when a user does a transaction, his/her user ID will be stored in the transaction model to reflect the change.

Program code 20 Database User Schema



In the Expenses tracker application, the Program code 20 user schema will hold all user-related information. This schema contains several fields, each having its type and attributes, as seen in the preceding figure. Name, email, and password are the three necessary fields that require an introduction.

Name: This field records the user's name who uses the application; its data type is String.

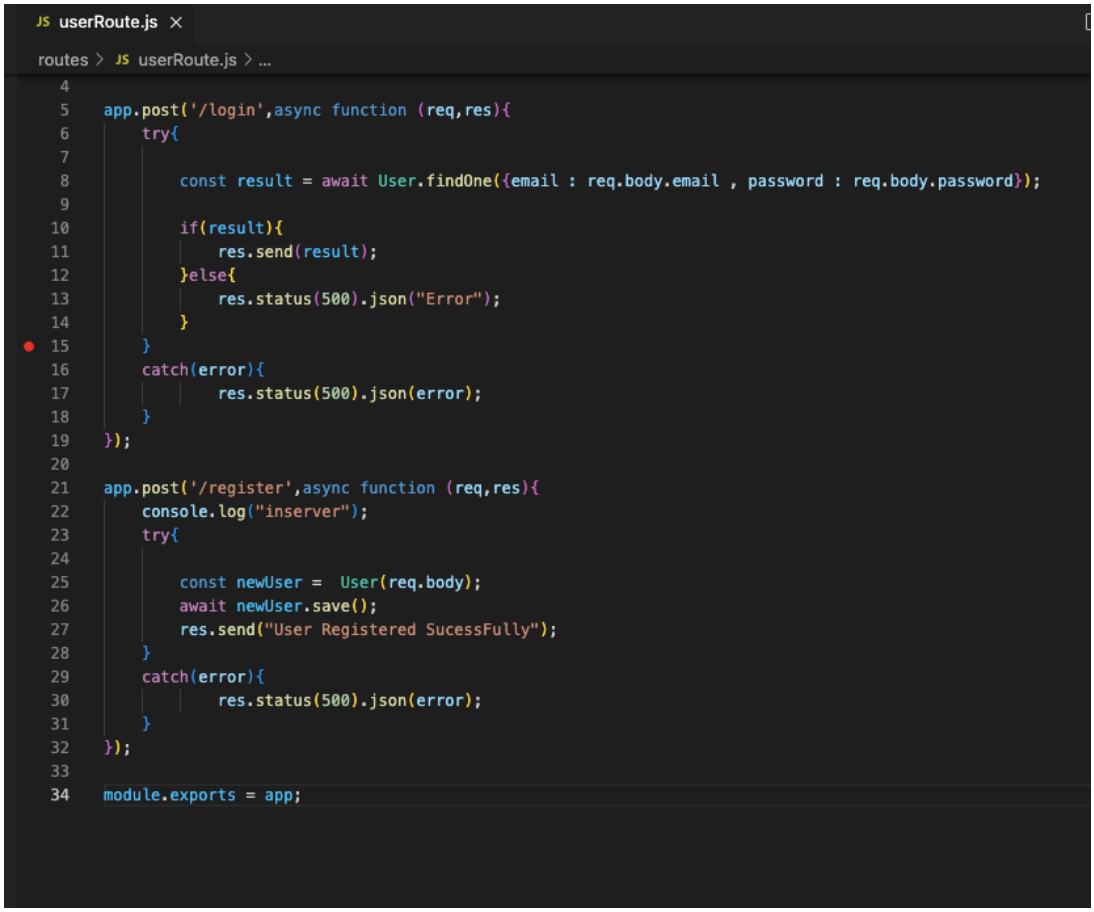
email: This field records the registered user's email address and is also of String data type.

Password: This field is intended to contain a password version of the user's password.

5.8.4 Configuring route in server side

Routing refers to how an application replies to a client request to a given endpoint, a URI (or path), and a particular HTTP request type (GET, POST, Etc.). A route is a request-handling function. This function requires two parameters, req, and res. A first parameter is an object containing request-specific information. A second parameter is an object containing response-related information.

Program code 21 implement User route in server side



As illustrate the Program code 22 is a javascript file that contains the code for the login and register routes. The login route takes in an email and password from the user, then checks if there is a matching record in the database. If there is, it sends back that record to the client. If not, it sends back an error message. The register route takes in all of the information from a new user and saves it to the database as a new record.

Program code 22 implement Transaction route in server side

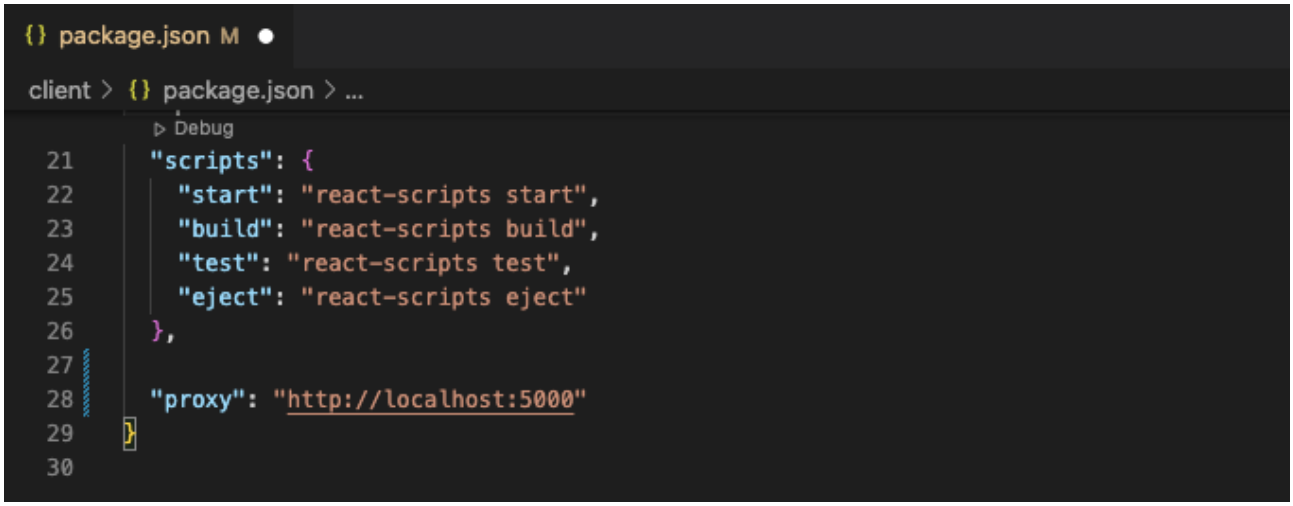
```
app.post('/add-transaction',async function (req,res){
    console.log("inserver");
    try{

        const newTransaction = Transaction(req.body);
        await newTransaction.save();
        res.send("Transaction Saved SucessFully");
    }
    catch(error){
            res.status(500).json(error);
    }
});
```

The function shown in Program code 22 that adds a new transaction to the database. It takes the request body and creates a new instance of Transaction with it, then saves it to the database. If there's an error, it sends back an error message in JSON format.

6 Connection between the front-end and the back-end.

Connect a React frontend to a NodeJS backend as a first step. Next, the react code must be configured to send fetch requests to the backend. Finally, both the frontend and backend code require a start script. In the terminal, you may do npm run start for react, and npm run server Start for NodeJS after installing node server.js. JSON, as seen below (if the programmer's main server file is called server.js). Additionally, a user must add a "proxy" entry to package.json that redirects fetch requests to the server's port. Therefore, if our NodeJS server operates on port 5000, the user must change the port number.

Program code 23 Connection between the front-end and the back-end with Proxy



The backend and frontend will not be able to communicate unless both are running; therefore, execute npm run start and node server.js in the terminal during the whole development process to ensure that your backend proxies may be received. Once these two are running, we can use React's useEffect hook to send a GET request to any named route.

7 Results

The thesis addresses the researched topics and might serve as a guide for developers, or anybody interested in developing an expense tracker application. The knowledge base facilitates comprehension of the most requested and highly valued features of an expense tracker app among users nowadays. Furthermore, the thesis helps to comprehend better how to make the app accessible to various users

Figma is a potent web prototyping tool compatible with all browsers and OS systems, and offline projects may be downloaded as files. It was tremendously valuable throughout the design phase of the practical implementation aspect of the prototype app. React, with its reusable components, was utilized to program the application's user interface. The framework's core included a virtual DOM program and server-side rendering, enabling the highly rapid execution of complicated applications. The capability to reuse components is one of the most significant benefits of utilizing React JS. It saves time since developers can build only a few scripts for equivalent functionalities.

This output of the thesis is a locally hosted front-end Expense Tracker application. The back end must be constructed for the application to function thoroughly. Following the conclusion of the thesis, more app development, notably for the hosted version, will be conducted.

The demo application can be found in the following GitHub repository:
https://github.com/arpit191001/Expense-traker-app-

8 Summary

Expense tracking has always fascinated me; thus, I chose to write my thesis on a topic linked to money tracking. Currently, the need for material based on expense trackers is increasing as a result of broad adoption. I was actually interested in creating my own expense tracker web application, so I based my thesis on that concept. The study issues were solved by assessing the requirements of a contemporary expense tracker application, which included certain functionalities. During the implementation phase, the practical development of the app's user interface with React and its backend with Node.js was resolved.

I got the opportunity to perform considerable study on a topic of great interest to me while writing my thesis. The prototype's implementation assisted me in understanding the process of utilizing open-source APIs with the help of the React frameworks.I'm excited to keep working on the project, give users more features, and finish the hosted version.

This thesis will likely serve as a reference in the future for individuals who wish to learn about expense tracking or how to construct a functional expense tracking app using React and Node.js.

References

Applause App Quality, Inc. (2022, October 19). *What is the European Accessibility Act?* Applause. https://www.applause.com/blog/what-is-the-european-accessibility-act

Buchenau, Z. (2020b, August 12). *10 Reasons Why You Should Track Your Expenses*. Be the Budget. https://bethebudget.com/reasons-to-track-your-expenses/

Bandakkanavar, R. (2022, July 11). *Software Requirements Specification document with example*. Krazytech. https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database

Beginning Node.js-Apress Download (297 Pages | Free). (n.d.-a). https://www.pdfdrive.com/beginning-nodejs-apress-d43544795.html

Botelho, B., & Vaughan, J. (2020, August 28). *MongoDB*. Data Management. https://www.techtarget.com/searchdatamanagement/definition/MongoDB

Duggal, B. (2021, September 15). *Developing an Expense Tracking App – Must-Have Features Your App Needs*. Mindinventory. https://www.mindinventory.com/blog/expense-tracking-app-development-features/

Duggal, B. (2021, September 15). *Developing an Expense Tracking App – Must-Have Features Your App Needs*. Mindinventory. https://www.mindinventory.com/blog/expense-tracking-app-development-features/

GeeksforGeeks. (2022, November 17). *Introduction to Visual Studio*. https://www.geeksforgeeks.org/introduction-to-visual-studio/

GeeksforGeeks. (2021, October 22). *What are the features of ReactJS ?* https://www.geeksforgeeks.org/what-are-the-features-of-reactjs/

Griffith, B. (2022, November 15). *Front End vs. Back End: What's the Difference?* Kenzie Academy. https://kenzie.snhu.edu/blog/front-end-vs-back-end-whats-the-difference/

GeeksforGeeks. (2021a, June 6). *What is MongoDB - Working and Features*. https://www.geeksforgeeks.org/what-is-mongodb-working-and-features/

Hámori, F. (2022, May 31). *The History of React.js on a Timeline*. RisingStack Engineering. https://blog.risingstack.com/the-history-of-react-js-on-a-timeline

Initiative, W. W. A. (n.d.). *Introduction to Web Accessibility*. Web Accessibility Initiative (WAI). https://www.w3.org/WAI/fundamentals/accessibility-intro/

Kai, B. (2022c, February 13). *Top 6 Features in React*. DEV Community 👩‍💻👨‍💻. https://dev.to/brayan_kai/top-6-features-in-react-3b18

Khizhniakova, M. (2022, February 21). *How an expense tracker can improve your business*. https://www.jenji.io/en-us/resources/how-an-expense-tracker-can-improve-business

Lane, K. (2021, April 8). *Intro to APIs: History of APIs*. Postman Blog. https://blog.postman.com/intro-to-apis-history-of-apis/

Perera, R. (2020, September 14). *What is Figma? (And How to Use Figma for Beginners)*. Theme Junkie. https://www.theme-junkie.com/what-is-figma/

ReactJS Features - javatpoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/react-features

Singhal, G. (2020, September 12). *Load NPM Modules with React*. Pluralsight. https://www.pluralsight.com/guides/load-npm-modules-with-react
https://www.geeksforgeeks.org/what-are-the-features-of-reactjs/

Savaram, R. (2022, May 25). *Introduction to ReactJS*. Mindmajix. https://mindmajix.com/introduction-to-react-js

The Design Project, & Flinn. (n.d.-b). *Application Design: Design, Build, and Implement | The Design Project*. https://designproject.io/blog/application-design

The Best Expense Tracker Apps for 2022. (2022, August 2). Investopedia. https://www.investopedia.com/best-expense-tracker-apps-5114591