Prologue

The PDF version of this file located at:

https://github.com/SPehsar/ExpenseTracker/blob/master/README.pdf

You can access the application using the following link:

https://expense-tracker-qa5s.onrender.com

Contents

Summary

Expense tracker app is an online app that allows you to log your expenditure and categorize your expenses to figure out the amount you spent or gained. It can provide information for certain periods and provide you with the net balance, as we all expect. The results appear in a tabular form accordingly. You can access the application using the following link:

How It Works

This app is a very simple version of budgeting, and it will provide users with a general view of incoming and outgoing money.

Authentication

Let's say kind of. You register and sign in first using your email address and password (**Please note: This is not a real secure place to use your actual email address or name. Make up something to practice and experience.**) You can log out at the end. The data will be saved on the MongoDB server.

Next time you sign back, the data will be available to you instantly. When you sign in, your name appears in the top-right corner. When you want to log out, you hover over your name and a drop-down box appears that has a "**Logout"** button to click on and log out.

When you are in the app

The first thing you want is adding a new transaction by clicking on the "**Add a Transaction button."** "**Add Transaction**" form appears that needs:

- Date of transaction
- Type of transaction
    - Income
    - Expense
- Amount of the transaction
- A description of the transaction, although it is not necessary.

All the fields are required, except the description part.

For this purpose, click on "**Add a Transaction"** button. You can cancel it by clicking on the "**Cancel"** button only. If you do not complete the require fields, clicking on the "**Submit"** button will not create any new transaction. The required fields should be completed for a transaction added.

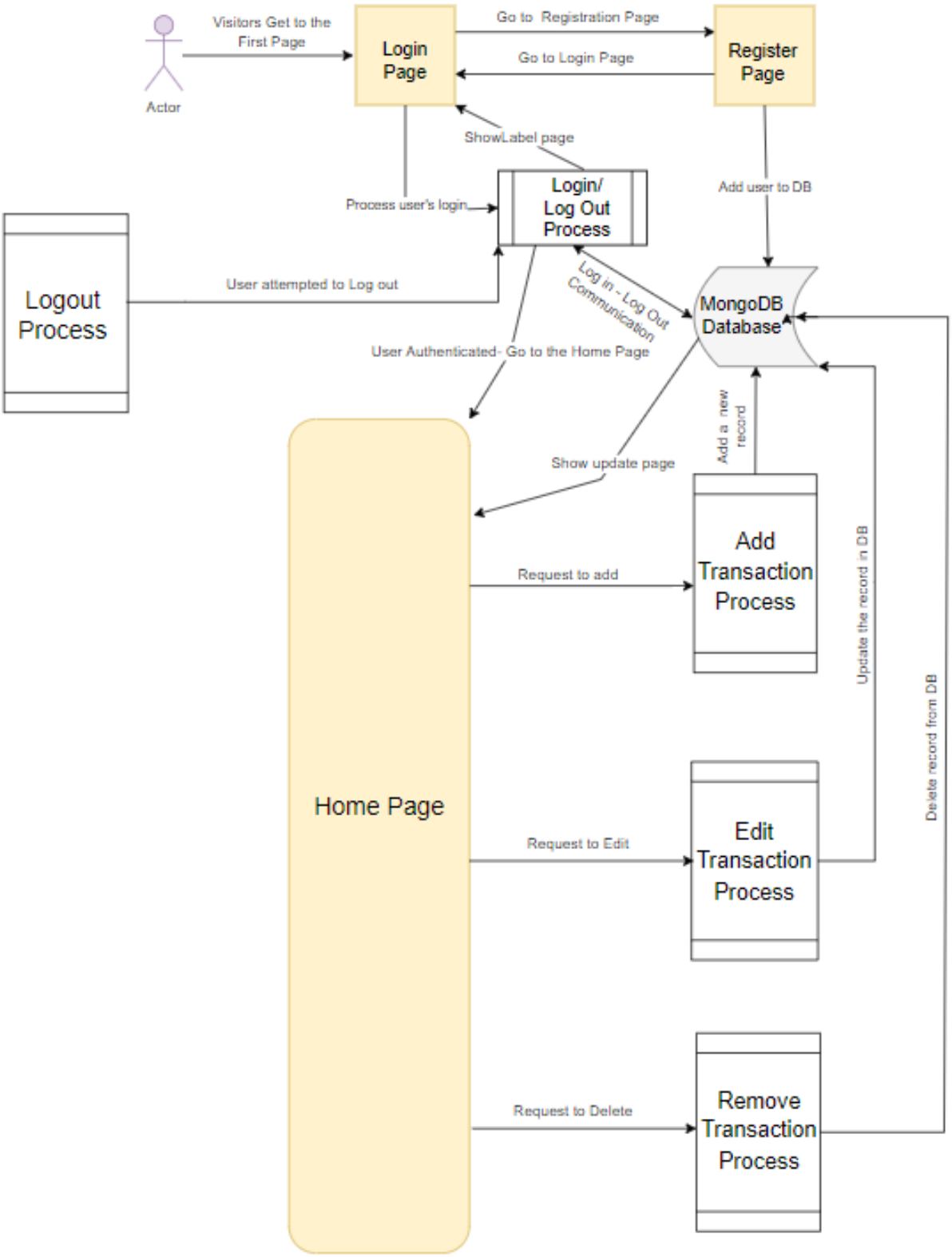
The transaction will appear on the screen according to its date, in date order. At any time, you can edit the transaction by clicking on the edit icon in the action column. You can also delete a transaction at any time by clicking on the delete icon in the transaction column.

There are two filters that help you choose what you want to see. These two, in combination, create a report. If the report is long, it will have several pages. There is an arrow (back and forth) to navigate through various pages of the report.
The two filters:

1. Select Report Type (Balance, Income, Expense)
    - Balance – Calculates total balance based on the target period.
    - Income only – Provide you with acquired income based on the target period.
    - Expense only – Provide you with the total you spend based on the target period.
2. Select Period:
    - Last 1 Week
    - Last one month
    - Last one year
    - Or you can customize it (viva to ANTD)

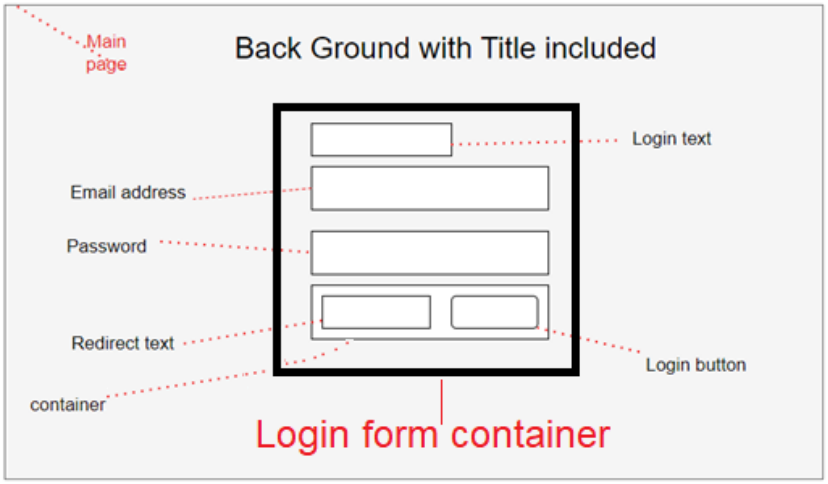
An analytical report appears on the right side of the screen, above the table, that shows information based on selected criteria from the above filters.

Communication Diagram

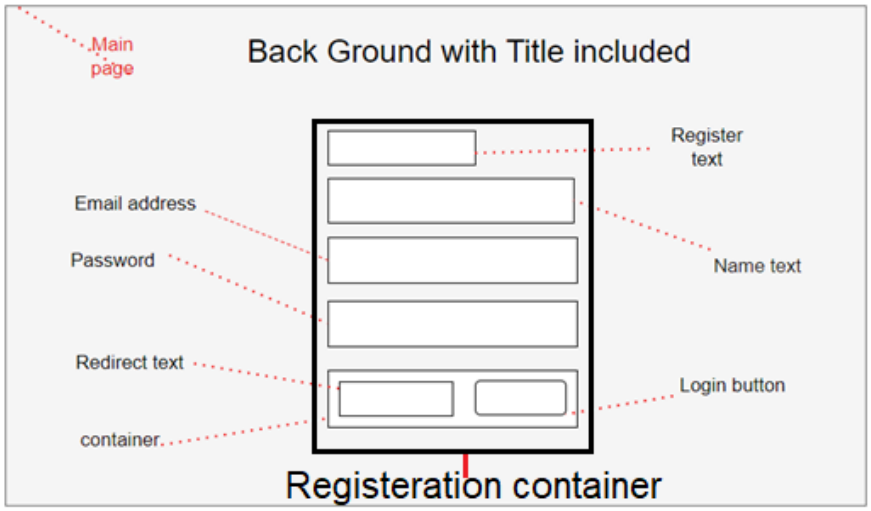

Actor
Visitors Get to the First Page
Login Page
Go to Registration Page
Go to Login Page
Register Page
ShowLabel page
Process user's login
Login/ Log Out Process
Add user to DB
Logout Process
User attempted to Log out
Log in - Log Out Communication
MongoDB Database
User Authenticated- Go to the Home Page
Add a new record
Show update page
Home Page
Add Transaction Process
Request to add
Update the record in DB
Delete record from DB
Edit Transaction Process
Request to Edit
Remove Transaction Process
Request to Delete

Wireframes

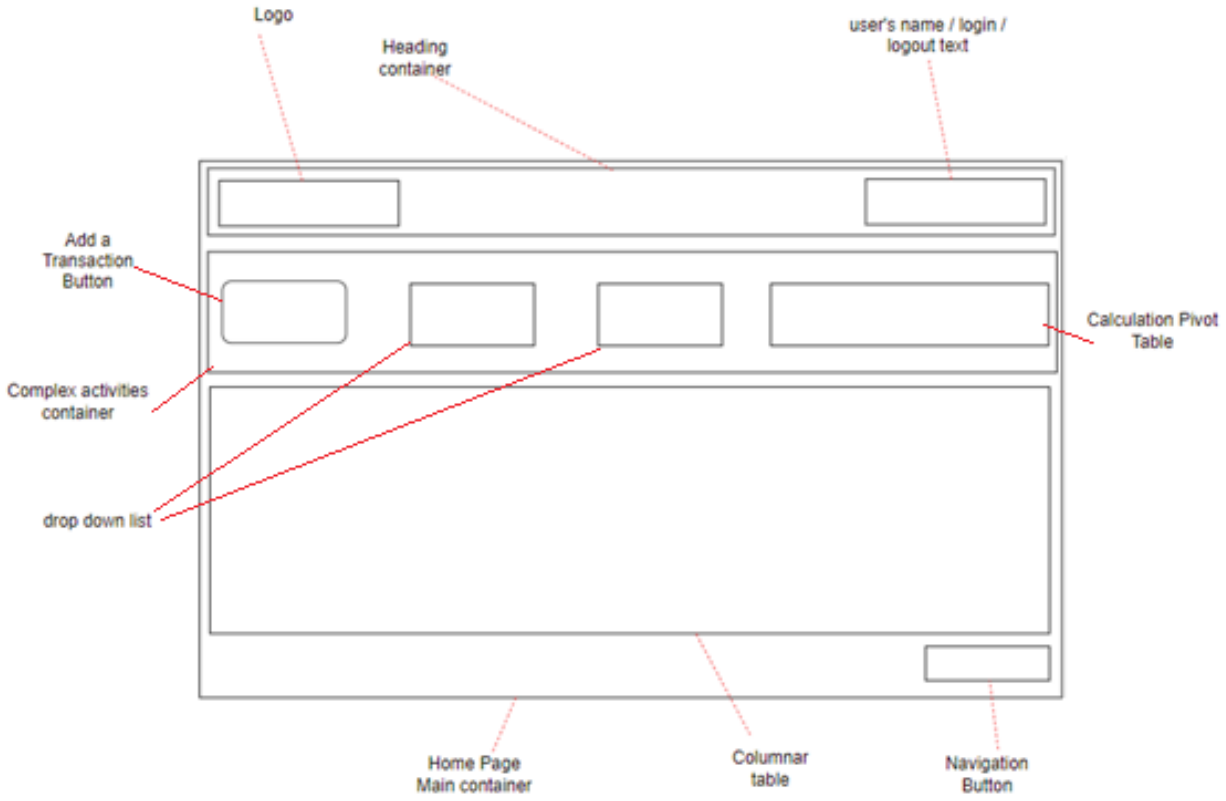
Login page



Registration page



Home Page



The Code

The following is the general structure of this expense tracking app. There are two parts involved (also including database):

- Front End
- Back end

User interface with the front end and behind the scenes gets connected to the back end. I used a mix of MongoDB, Express, React, and NodeJS with a nudge toward HTML, CSS, and Windows environments to deliver this app.

The technology is being used

The four technologies that make up the technology stack are all JS-based, so we use JavaScript for coding too.

Node JS is a JS runtime environment, i.e., it enables running JavaScript code outside the browser. User interfaces with the front end and behind the scenes and gets connected to the back end. I used a mix of MongoDB, Express, React, and NodeJS with a nudge toward HTML, CSS, and Windows environments to deliver this app.

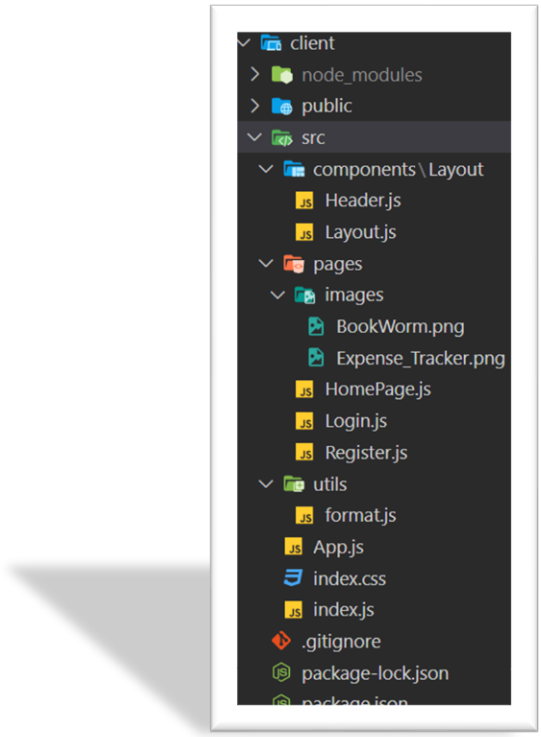
This way, we can easily construct a three-tier architecture (front end, back end, database) entirely using JavaScript and JSON. React handles the front end, while Node and Express deal with the back end. MongoDB is the database of choice in this stack.

Developers need to create an account to be able to use a database server for saving data. It is a free account if you are willing to use it.

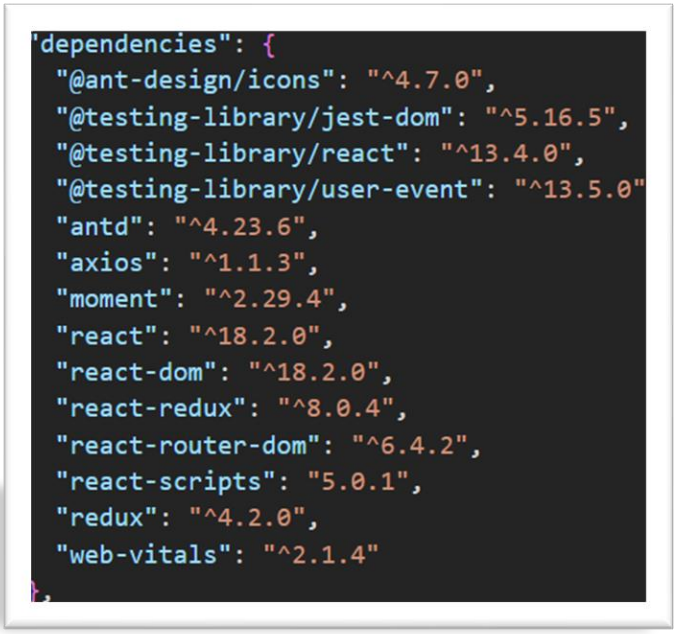
I have used VS Code for the coding.

Front End

The following shows the structure of the front-end part of this app.

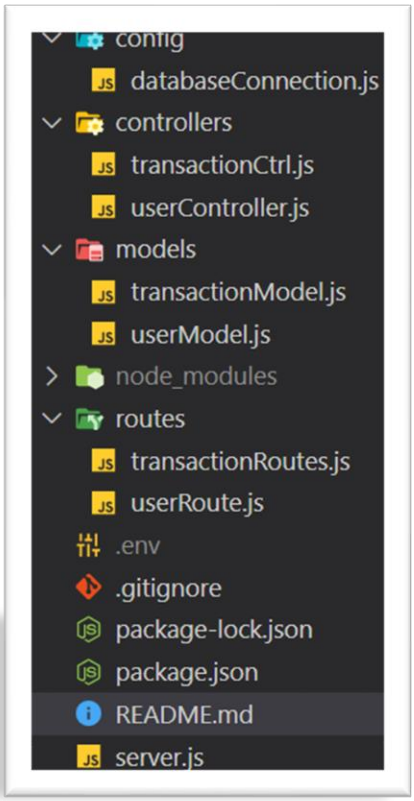


I have used followings as dependency an installed them using "**npm i**" command in the VS Code console. **To run Front End only**, in console, "**cd client**", to make sure you are in the client folder. Then run "**npm start**" command.



Back End

Back-End part has the following structure:



I have used followings as dependency an installed them using **npm i** command from VS Code console:

To **run Back End only**, in console, "**cd ..**" to come out of client folder if you were in it following above instruction, and make sure you are in the root folder of project. Then run "**nodemon server**" command.

To **run application concurrently** from the VS Code console, make sure you are in the root folder of project, Then run "**npm run dev**" command.

How I Created the Structure

In VS Code, from command console:

Back end:

Create all the needed back-end folders and files
Go to console and type: npm init -y
(it creates package.json)

Then install all dependencies for back end using npm i

Make sure to create .env and .gitignore files.
Include /node_modules in .gitignore file to prevent uploading those large files/folders to github.

Front end:

npm create-react-app client
cd client
Then install all dependencies for front end using npm i
Create all the needed folders and files and make sure you have .gitignore file.
Include .env and /node_modules in .gitignore file to prevent password uploaded to github.

You may go ahead and clean up some folders and files in front end part. They are created by default and if you don't use them remove them

Final Note

The code was created based on the knowledge acquired during cohort class, special the full stack MERN app that we worked on. In addition, shout-out to al youtubers and other coders that provide very detailed educational instructions. You all rock with no exception. Besides that, If you like the code, please give me a star.

I, definitely, suggest that you explore "antd" (see below for more information). They are resourceful in all aspects.

Need to Know

Followings are quotes that I have grabbed "word by word" by googling. I have not the written any of these; but they provide ample information if you want to practice and learn with my code.

Moment.js

A JavaScript date library for parsing, validating, manipulating, and formatting dates which is freely distributable under the terms of the MIT license. So, what does ***moment*** do?

Moment JS allows displaying of date as per localization and in human readable format. You can use Moment JS inside a browser using the script method. It is also available with Node.js and can be installed using npm. (https://www.npmjs.com/package/moment)

Ant Design

An enterprise-class UI design language and React UI library. It can be installed, using npm i antd.

Features

Ant Design is an open-source design system developed by the Ant Group–parent company of Alibaba, Alipay, Huabei, and MYbank, to name a few. The component library supports React, Vue, and Angular front-end frameworks.

- An enterprise-class UI design system for web applications.
- A set of high-quality React components out of the box.
- Written in TypeScript with predictable static types.
- The whole package of development and design resources and tools.

People contribute to it. It is part of the Open collective. (https://ant.design/)

Redux

Redux, meaning 'brought back, restored' (from Latin reducere, 'to bring back'), is a predictable state container designed to help you write JavaScript apps that behave consistently across client, server, and

native environments, and are easy to test. While it's mostly used as a state management tool with React, you can use it with any other JavaScript framework or library.

Now, is Redux front end or back end?
It should be clear that Redux can be used for the client side (frontend) with user interfaces. However, since Redux is just JavaScript, it can also be used on the server side (backend).

Axios

Axios allows us to communicate with APIs easily in our React apps. Though this can also be achieved by other methods like **fetch** or **AJAX**; but Axios can provide a little more functionality that goes a long way with applications that use React. Axios is a promise-based library used with Node.

As a library it serves to create HTTP requests that are present externally. It is evident from the fact that we may sometimes in ***React applications need to get data from the external source***. ***It is quite difficult to fetch such data so that they can be normally shown on the website.***

Axios vs. Fetch

That is being said, so what is the difference between Axios and fetch?

Different properties are used for a post request to send data to the endpoint - *Axios uses the data property*, *whereas with fetch we use the body property*. **We need to serialize data into a JSON string to send data**. **Axios automatically stringifies data** *when* ***sending*** *JavaScript* ***objects to the API using the POST*** *method.*

CORS

Cross-Origin Resource Sharing (**CORS**) is an HTTP-header based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources. Now you may wonder if CORS is front end or back end.

CORS or "Cross-Origin Resource Sharing" refers to the situations when a front end running in a browser has JavaScript code that communicates with a backend, and the back end is in a different "origin" than the frontend.

Concurrently

Concurrency means that a program is able to run more than one task at a time (one command inside the same terminal). Generally, in order to run your React app with your Node Express API back end on your localhost server, you would usually need to run commands like npm start for both the front end and the back end. [I used npm run dev that was defined in the package.json] . By the way, this is not to be confused with parallelism. During concurrency, different tasks with differing goals of the program can be performed at the same time, while in parallelism, different parts of the program execute one task.

Morgan

Morgan is an HTTP request level Middleware. It is a great tool that logs the requests along with some other information depending upon its configuration and the preset used. It proves to be very helpful while debugging and also if you want to create Log files. Prerequisites: Basic understanding of Nodejs.

Happy coding!