Contents

Introduction

My bookkeeping application is targeted at small business owners or sole traders with a fundamental understanding of accountancy. Users who don't need many of the features that clutter the UI and confuse the customer like that which is offered by my competition such as Xero, FreshBooks and QuickBooks. The application is a practical and straight forward alternative that gives the user full control over which ledgers they wish to make double entries into. This means they can create their own accounts in a transparent manner by being able to directly see the T-accounts that they are making entries into as well as edit or delete those entries. The necessary financial reports such as the Profit and Loss statement and the Balance sheet are generated automatically and overview reports such as bar and pie charts help to give a quick financial overview of the business. The application gives what is needed to the customer is a clear manner compared to the competition where users may need training to operate it [19].

What makes this project unique is its simplicity and transparency as users may edit and delete records and are also not overwhelmed by large numbers of features. Users can view ledgers in the T-account format that they are taught when learning accounting which is completely familiar to them to avoid confusion. When making journal entries they also have full control over which accounts they can make entries into.

My aim is to build a web application to support the basics of accountancy. The application will allow bookkeepers for small businesses to input and track their business transactions for accounting purposes. For example, their income, expenses, assets, liabilities. The application allows them to get an idea of where their business stands financially during their financial year by generating financial statements, namely the statement of financial position & the statement of profit and loss.

Design & Implementation

Requirements

Account Features

- Users must be able to create an account.
- Users must be able to record the key business information. (Company name, address, telephone, base currency, financial year end date).
- Users must be able to log in.
- Users must be able to log out.

Recording Financial Data

- Users must be able to create journal entries to record their business transactions.
- The system must provide a list of default nominal accounts that the user can select when entering journal transactions.
- Users must be able to delete or edit journal transactions.
- Users must be able to record payments & receipts of the business.
- The system must generate unique references for each financial transaction.

Viewing Financial Data

- System must display all the ledgers used and their transactions and balances.

- System must be able to generate the Profit & Loss statement and the statement of financial position.

Overview

My application has been designed using a client-server model. My frontend makes API requests to backend to either create, read, update, or delete data. My backend is a REST API that will interact with my database which is built using SQL.

Frontend

Next.js

The frontend is built using Next.js which uses React.js and server-side rendering. React allows me to build the application using reusable components. Server-side rendering is when HTML is generated on the server and then sent to the client. This provides a faster loading time for the client, and it also improves search engine optimization for my website.

Tailwind CSS

I also used the Tailwind CSS framework to improve the speed at which I could build the UI and write CSS. Tailwind allows you to write CSS directly into you HTML by using utility classes. This saves time as you don't need to create separate CSS files for each HTML page but instead select the classes you need directly.

Redux.js

Redux is a state management tool that allows you to save data in a store so that any component from any page can access it. It makes it easier to pass data between components and pages in React.js without having to bubble up.

Backend

Node.js

Is a runtime that allows me to build my backend using JavaScript. This means I will only need to use one programming language to build both my front and backend. It also allows me to use many different packages from the node package manager such as Express.js and Sequelize.js.

Express.js framework

Will allow me to create a REST APIs server that will accept requests from the front end and return the data required.

Sequelize.js

Is the Object Relational Mapper that will allow me to create my database tables with relationships and query them. It reduces development time as I don't have to write the specific SQL queries but can call methods instead.

Security

When researching the latest security techniques for web applications in industry, I decided to use JWT for authentication and authorization of users. Once a user has signed into my application, I will send them an encrypted token that contains their user information and credentials. Whenever a

user communicates with my server, I decrypt this token to verify their identity and then modify their records as requested.

I decided to use Bcrypt for encryption of passwords on the database as it has not been broken and is very strong and slow encryption [22].

I intend to setup HTTPS in the future when my application is hosted on the cloud. HTTPS is the secure version of HTTP which is the protocol that is used to send data across the internet [23]. This means that whenever a user logs into my application or sends sensitive data it will be encrypted.

Database

The database that I use is PostgreSQL since it is the most suitable for the type of data I am inputting. A business's financial data is highly structured, so it fit well in a SQL database using tables. It would not be suitable for a key to value relationship used in no-SQL databases. PostgreSQL is ACID compliant which is useful for transactional data involving double entry. It is also easier to work with and I prefer their documentation.

Schema

Transactions

The transactions table will be used to store each business transaction. Each transaction would be categorised as either a debit or credit using the entry type column and have a nominal account to add further detail to the type of transaction made.

Nominal Accounts

The nominal accounts table will be used to categorise each transaction. Some example nominal account names will be stationary, fuel, sale of assets, income. They will have a reference number to further categorise each transaction for reporting purposes. For example, all nominal accounts referenced 5000 to 6000 are expenses. This also allows the possibility for the user to create their own nominal accounts in the future as they could create a custom name and use a reference number within the specific range to indicate what type of category the transaction would be.

Business Data

This table will store all the general data about the company.

Users

This table is to store the information about the user such as email and password. This table will be queried during authentication.

Testing

Figure 28: Postman API test for login route

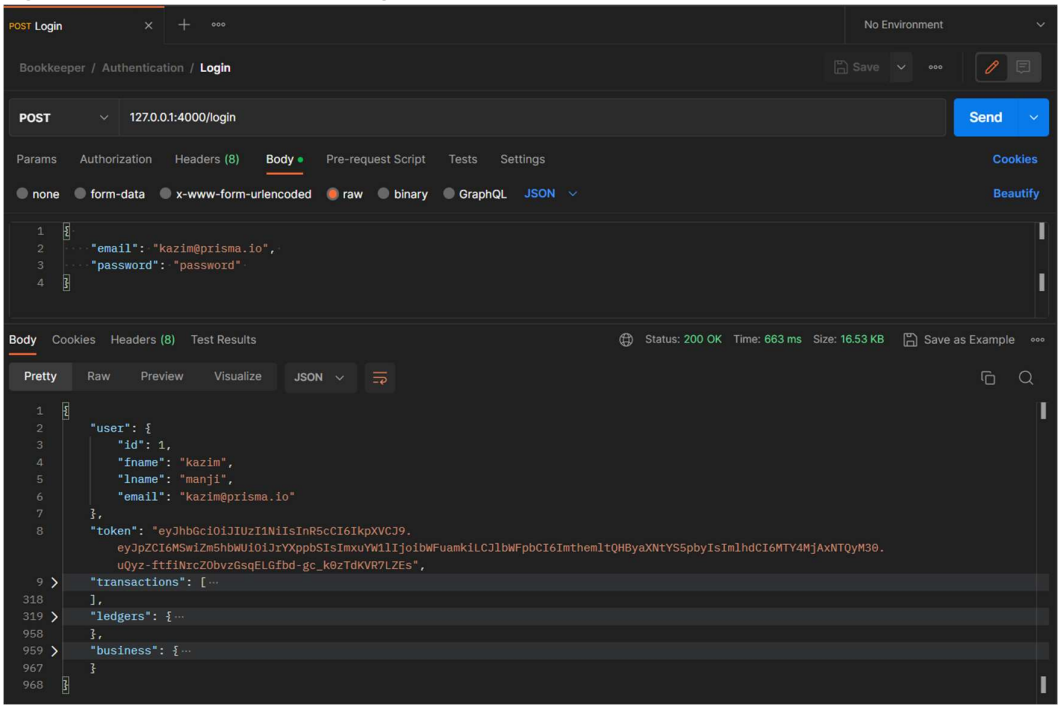
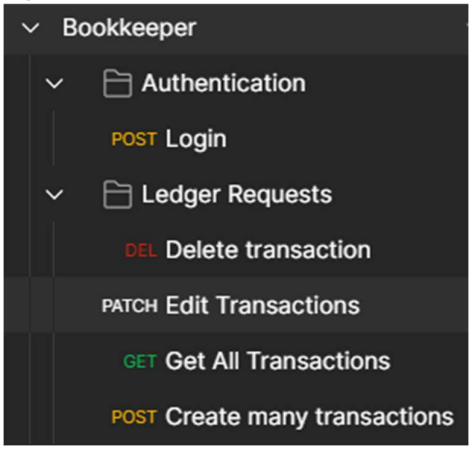


Figure 29: Postman API tests



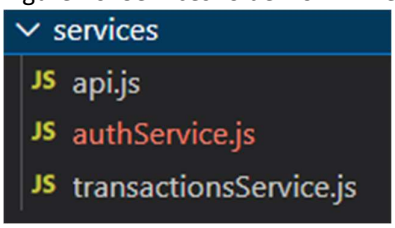
I performed API testing using a software called Postman where I was able to send requests to my backend and view the return JSON objects [24]. Figure 28 gives an example of the test for my login route. I wrote tests for all the communications between my frontend and backend as shown in figure 29. While writing my code I also utilized try and catch statements that allowed me to return any errors during my development. This meant that my console would be displaying any errors if they occurred. Using try catch statements also meant that my server would not crash if an error occurred which meant a user could continue to use the application.

Project Structure

Frontend

Services

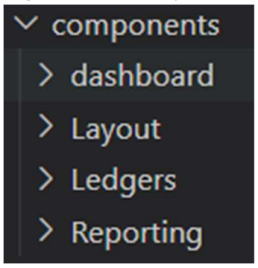
Figure 20: Services folder for API requests



I used the Axios library to make API request from my frontend to my backend. These requests are made up of authentication requests such as login, register, logout which are stored inside authService.js and transaction requests such as create, edit, delete transactions which are stored inside transactionService.js. I initialized the Axios instance inside api.js and set the JWT authorization header by reading from local storage if the user has logged on. I then created functions for each request in separate files for code readability.

Components

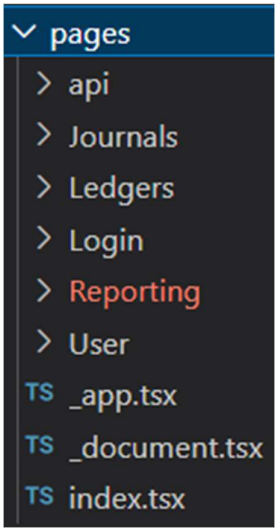
Figure 21: Components folder



I have created React.js components that are used by different pages and stored them in this folder. My pages can import these components when needed.

Pages

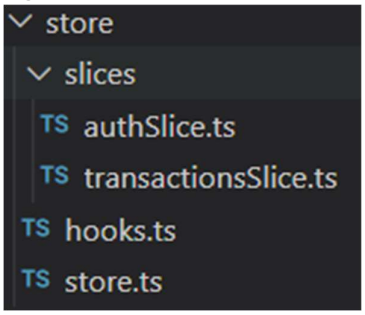
Figure 22: Pages folder



Under each subfolder is the React component that makes up each page in my website. Each folder here is associated with a different route for example the component returned in the /Journals route will be the one inside the Journals folder here. Some of these pages import components from the component folder. Index.tsx is my dashboard landing page.

Redux Store

Figure 23: Redux folder

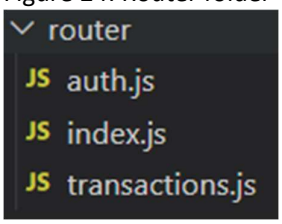


The redux store is used to persist the application state between pages. It is like a database and allows me to access and mutate data from different pages and components. The redux instance is initialized inside store.ts and the application specific services are inside the slices folder. When I make requests using Axios the data is stored here. Whenever I refresh a page, my redux store will check if the data already exists in local storage and save it. This prevents my application from loosing all data upon refresh.

Backend

Router

Figure 24: Router folder



My server listens for requests coming to it using Express.js. These listeners are implemented in the router folder in different files for readability. They call middleware functions as well as controller functions.

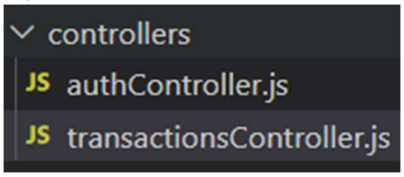
Middleware

Figure 25: Middleware folder

Middleware functions can be chained inside of my Express.js listeners. Once a request is received, I can pass it on to multiple middleware functions before sending it to my controller. The auth.js middleware function is used inside my routes so that my server will check the authenticity of the request before processing. The function will decrypt the JWT token found in the request header and get the user's information and pass this information along with the original request to my controller. If the token is expired, the request will be rejected.

Controllers

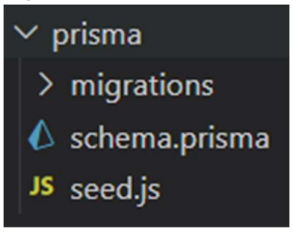
Figure 26: Controller folder



The controllers will perform CRUD operations on my database as requested by the client. They are encapsulated in try and catch statements to prevent my server crashing if there is ever an issue with a request and instead return an error message. This was challenging at times as I had to learn about concurrency in JavaScript and how to use “async”, “await” and “then” statements allow database operations to be performed asynchronously. This meant my server could wait for database operations to finish before moving on to the next line code.

Prisma ORM

Figure 27: Prisma folder



Prisma is the object relational mapper, and this folder contains all the code that allows me to create database tables and interact with them. My controller will import code from here to interact with the database. The seed.js file allows me to populate my tables with test data. The schema defines my database tables and their relationships. The migrations folder keeps track of any database migrations and changes occur.

Critical Appraisal

Summary of work done & Critical analysis of work done:

I believe that the original aims of the project have been achieved as I have bult a full stack web application that allows users to record their business transaction following the double entry bookkeeping rules. The users have the freedom to select which ledgers they want to make records in and can delete or edit those records, meaning they have full control over their accounts. The users can use the overview reports in the dashboard page but also view their Profit & Loss statement and Balance sheet.

I believe the UI is simple to use which is part of my unique value proposition, but it could be improved by writing a user guide on how to use my application. Tutorial videos could also be created and would also act as advertising content if posted to social media. I believe the ledgers page could have more filters to manage each transaction. Further testing could be carried out such as compatibility with different browsers. Usability testing would also have been beneficial for allowing me to get direct feedback from users.

There are some enhancing features that could be included to improve the application and make it more user friendly. For example, If the user could download the reports as a PDF file, it would allow the user to share those documents with the necessary parties in the organization. I could also improve the dashboard reports by building a regression model to be able to predict the users cashflow for the future and presenting it as a report to the user [27]. By enhancing my application through new features that were not listed in my requirements I would add more value for the customer and would be more likely to get sales.

Discussion on Social, sustainable, commercial, economic context

The benefits for businesses would be that they would be able to use my application to record their transactions and better understand their financials. They can examine the reports to get an overview on their expenses better manage their finances by understanding their fixed and variable costs. By examining their transactions, they can make estimates on the future and make better management decisions.

Hacking and data loss is potential risk as this sensitive business data could be stolen by hackers. The use of any new software also means that users must be trained and get used to doing things different. My business is intended solely for commercial purposes, but it can be used for learning by those new to accounting. They may better understand bookkeeping and practice making journal entries to build a profit and loss statement or balance sheet.

Since I will be competing with other businesses in my industry, I will be pushing them to become more competitive and could drive innovation to even create new products or features [25]. They may need to hire new developers to achieve this which creates jobs. New markets could be created through innovation which could grow the economy by creating new job opportunities and revenue streams [25]. Generally, competition is good for consumers as it causes businesses to lower prices and build better products [25].

Personal development

Throughout the project I have been able to improve my understanding with JavaScript and learnt how to incorporate TypeScript for the first time which provided my frontend with type safety. I have used new frontend frameworks such as Next.js and Tailwind.js to build my application in a component-based manner. I had researched into web security to learn about HTTPS, encryption algorithms and authentication using JWT. I improved my problem-solving skills as I ran into many issues when trying to work with different libraries that I had not used before. I would carry out research and come up with workarounds to resolve those problems allowing me to apply myself fully. I have gained a better understanding over asynchronous code and the difference between lazy and eager loading of database entities to be used in my backend. Overall, I have a better understanding of full stack web applications and the different methods to build them.

I learnt how to analyse competitors and build a lean canvas to better understand the components that make their business. I also applied this to my own business and was able to understand how to differentiate myself my developing a unique value proposition. I also learnt how to make estimates of expenses and revenues and plan different marketing strategies to suite my business. I learnt how to identify potential business partners and understand their benefits. I learnt how to specify my customer segment and research where to find potential customers and potential advertising platforms to target them.