Bookkeeping Web App for Information Organization

Github Repo | Demo

I. Introduction

Managing personal finances is a crucial yet often time-consuming task. As an international graduate student relying on family support, I need to monitor my monthly expenses and occasional income sources—such as fellowships, promotion rewards from banks, and GSI salaries—for effective financial planning. This is not only essential for my own budgeting but also helps my family stay informed about my spending in the United States.

To address this challenge, I developed the Bookkeeping Web App, which automates the organization of financial data from receipts and invoices. The app uses Veryfi's OCR API, a tool specialized in processing receipts and invoices, to extract numerical and textual information from uploaded images. Users can categorize the extracted data as either income or expense, and the app stores it in MongoDB for future retrieval and visualization.

This project integrates concepts of categorization, schema development, and retrieval mechanisms discussed in the course, applying them to a real-world scenario.

II. Technical Implementation

1. Technology Stack

The **frontend** was developed using **React.js**. Key libraries include:

- styled-components: For component-based styling.
- moment: To handle dates and times in the categorization process.
- react-chartjs-2: For creating dynamic visualizations of income and expense trends.
- react-datepicker: For date selection during data filtering.

The **backend** was built with **Node.js** and **Express**. Core tools include:

- mongoose: To interact with the MongoDB database, enabling schema design and data management.
- axios: To handle API requests to Veryfi and communication between the frontend and backend.

The OCR functionality relies on **Veryfi's OCR API**, which processes receipt images to extract structured data, including transaction amounts, dates, vendor, and descriptions.

Data is stored in **MongoDB**, offering scalability and flexibility for categorizing and retrieving transactional data.

2. System Workflow

a) Image Upload and OCR Processing

Users currently input the URL of a receipt image into a designated field. Upon clicking the submit button, the app sends the URL to the backend, which forwards it to the Veryfi OCR API for processing. Parsed data is returned for categorization.

b) Categorization and Tagging

If the user enters the input manually, they will be required to specify the entry as either income or expense, the amount, the date, the source or usage, and an description. Otherwise, these information is extracted from the receipt image.

c) Data Storage and Retrieval

Categorized transactions are stored in a MongoDB database, with each record assigned a unique identifier. This allows for efficient retrieval and modification.

d) Visualization

Using react-chartjs-2, the app generates visual representations of financial data, providing insights into income and expense trends.

III. Integration of Course Concepts

1. Categorization and Schema Development

The app's categorization system reflects the course concepts of schema design and data organization. Transactions are structured as a schema with predefined fields, including title, amount, date, category, and description. Each transaction is automatically assigned a unique identifier (id) upon creation in MongoDB. This schema not only ensures consistent data storage but also facilitates future filtering and searching functionality.

2. Information Retrieval Mechanisms

Efficient retrieval is a cornerstone of the app's design, achieved through the consistent use of the predefined schema:

The frontend interacts with the backend via RESTful APIs, exchanging data formatted according to the schema. For example, when submitting a transaction, the frontend sends a structured payload containing the transaction's title, amount, date, category, and description.

The backend communicates with MongoDB using mongoose, which enforces the schema during data creation and retrieval. When users request specific records (e.g., by ID), the backend uses MongoDB's API to locate and return data efficiently.

By adhering to a unified schema across the system, the app ensures that data remains consistent and queryable, laying the groundwork for implementing advanced features like filtering and searching. This design aligns with the course's emphasis on structured information organization and retrieval mechanisms.

IV. Significance and Future Improvements

The Bookkeeping Web App simplifies financial data management for individuals like me who need a streamlined solution for tracking expenses and incomes. It saves time, reduces manual effort, and enhances data accuracy.

Potential areas for improvement include:

1. User Authentication: Implementing a user login system and integrating Facebook and Google account APIs for seamless authentication and personalized user experience.
2. Enhanced Categorization: Implementing two-level categorization (e.g., "Food" → "Dine Out"/ "Groceries") and allowing users to customize categories by adding, editing, or deleting them.
3. Faceted Search: Search in all fields or a specific field to filter transactions exclusively by income or expense type.
4. Drag-and-Drop Image Upload: Allowing users to upload receipt images via drag-and-drop functionality, making the app more user-friendly.
5. Editable Parsed Results: Displaying the parsed results in a form where users can review and edit the extracted data before saving it.
6. Mobile App: Expanding the platform to mobile devices for on-the-go accessibility.

7. Advanced Analytics: Integrating predictive analytics to suggest budgeting strategies based on spending patterns.

V. Attribution

1. Veryfi OCR API for extracting data from receipts. https://www.veryfi.com/receipt-ocr-api/
2. React.js, Node.js, MongoDB, and associated libraries for development.
3. The frontend is hosted at Netlify, and the backend is hosted at Heroku.
4. References to tutorials and example code:

Tutorial 1

The Code Dealer. (2023). *Full Stack Development With React And NodeJS Expense Tracker Application* [Video]. YouTube. https://youtu.be/i0JesTevAcA?si=coxF5topw3xcSYfJ

Github Repository

Maclinz. (2023). *expense-tracker fullstack*. GitHub. https://github.com/Maclinz/expense-tracker_fullstack

Tutorial 2

Lets Build Together. (2023). *Build Your Receipt Scanner with Node.js - Step by Step Tutorial* [Video]. YouTube. https://youtu.be/0Zn-MmNmpJU?si=eZabfdKZq17C4Ddz

Image

Peiseka. (n.d.). *[Background]*. Retrieved from https://peiseka.com/Upload/mihuan/thumb/mihuan66.jpg

Flaticon. (n.d.). [User Icon]. Retrieved from https://www.flaticon.com/free-icon/user_1177568?k=1733895583641