

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	2
	LIST OF TABLES	3
	LIST OF FIGURES	4
1	CHAPTER 1	5
	INTRODUCTION	
	1.1 HTML	5
	1.2 CSS	5
	1.3 JavaScript	5
	1.4 MEAN Stack	5
2	METHODOLOGY	6
	2.1 Objective	6
	2.2 Problem Statement	6
	2.3 Block Diagram	7
	2.4 Module Explanation	
3	RESULTS AND DISCUSSION	7
4	CONCLUSION	8
5	SCREENSHOTS	8
6	REFERENCES	9

ABSTRACT

1) BudgetPal: A Personal Finance Management System offers a comprehensive solution for managing personal finances through a user-friendly interface built with React.js. This intuitive frontend allows users to effortlessly input financial data and access detailed reports of their activities.

2) The application enables users to categorize and monitor expenses, track multiple income streams, and manage their budgets effectively. With real-time data analysis and graphical insights, users can easily identify spending patterns and optimize their financial decisions. Data security is ensured through MongoDB, providing encrypted and efficient storage solutions.

3) Additionally, the scalability of the MERN stack allows the application to grow with an increasing user base and large datasets. By integrating these features, BudgetPal demonstrates the power of full-stack development in addressing real-world finance challenges, offering a secure, scalable, and efficient personal finance management tool..

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
1	Introduction	5
2	Methodology	6
3	Result and Discussion	7
4	Conclusion	8
5	Screenshots	8
6	References	9

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1	Main Page	8
2	Database Stored	9

CHAPTER 1

INTRODUCTION

1.1 HTML

HTML (HyperText Markup Language) forms the foundational structure of the **BudgetPal** web application. It defines the layout of elements such as input fields, buttons, and navigation menus for the user interface. In this project, HTML ensures that users can easily navigate through the application to manage their personal finances.

1.2 CSS

CSS (Cascading Style Sheets) is responsible for the visual appearance of the application, providing styles for layout, and colors. In **BudgetPal**, CSS ensures that the interface is clean, user-friendly, and responsive, adapting seamlessly across various devices.

1.3 Java script

JavaScript adds interactivity to the **BudgetPal** application, enabling features like dynamic updates of financial data, and form validation. JavaScript plays a crucial role in enhancing the user experience by allowing users to track expenses, set budgets, and visualize financial insights without reloading the page.

1.4 MERN Stack

- **MongoDB:** Stores user data such as income, expenses, and budget records.
- **Express.js:** Manages server-side logic, handling API requests, and routing.
- **React.js:** Builds the dynamic and interactive frontend for managing personal finance.

- **Node.js:** Processes backend operations and connects the server to the database.

CHAPTER 2

METHODOLOGY

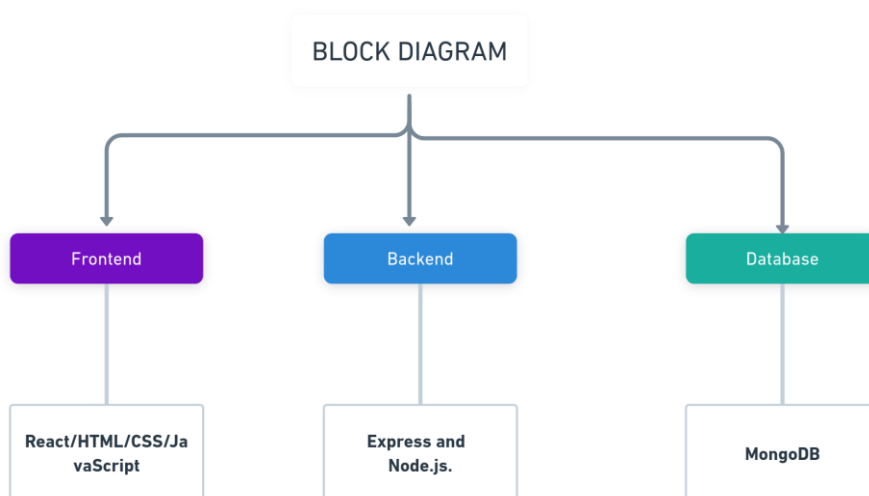
2.1 Objective

The goal of BudgetPal is to create a comprehensive personal finance management system that allows users to track their income, expenses, and savings. The application provides an intuitive platform where users can set financial goals, analyze spending patterns, and manage budgets effectively.

2.2 Problem statement

Managing personal finances can be overwhelming without the right tools. Many users struggle to track expenses, maintain budgets, and save efficiently. **BudgetPal** aims to resolve this issue by providing an easy-to-use platform that empowers users to stay on top of their financial health, make informed decisions, and ultimately improve their budgeting habits.

2.3 Block Diagram



2.4 Module Explanation

- **Frontend Module:** Handles user interactions, such as inputting financial details and generating reports using React.js for real-time data updates.
- **Backend Module:** Uses Node.js and Express.js to manage data requests, ensuring secure and accurate data processing.
- **Database Module:** Stores users' financial history, budgets, and spending patterns, allowing for comprehensive data management and reporting.

CHAPTER 3

RESULTS AND DISCUSSION

3.1 Result

The **BudgetPal** web application successfully enables users to manage their personal finances with ease. Users can input financial data, generate real-time reports on their spending, and adjust budgets accordingly. The application also provides graphical representations of financial data to help users visualize their progress towards savings goals.

3.2 Discussion

- The use of the MERN stack was instrumental in creating a robust, scalable web application. MongoDB provided a flexible way to store unstructured financial data, while Express.js and Node.js enabled seamless communication between the frontend and backend.
- React.js ensured that the user interface remained responsive and dynamic. Overall, the **BudgetPal** project demonstrated how the MERN stack can efficiently handle the complexities of personal finance management.
- For future iterations, the application could benefit from features like automatic expense categorization, personalized financial recommendations, and integration with bank APIs for real-time transaction tracking.

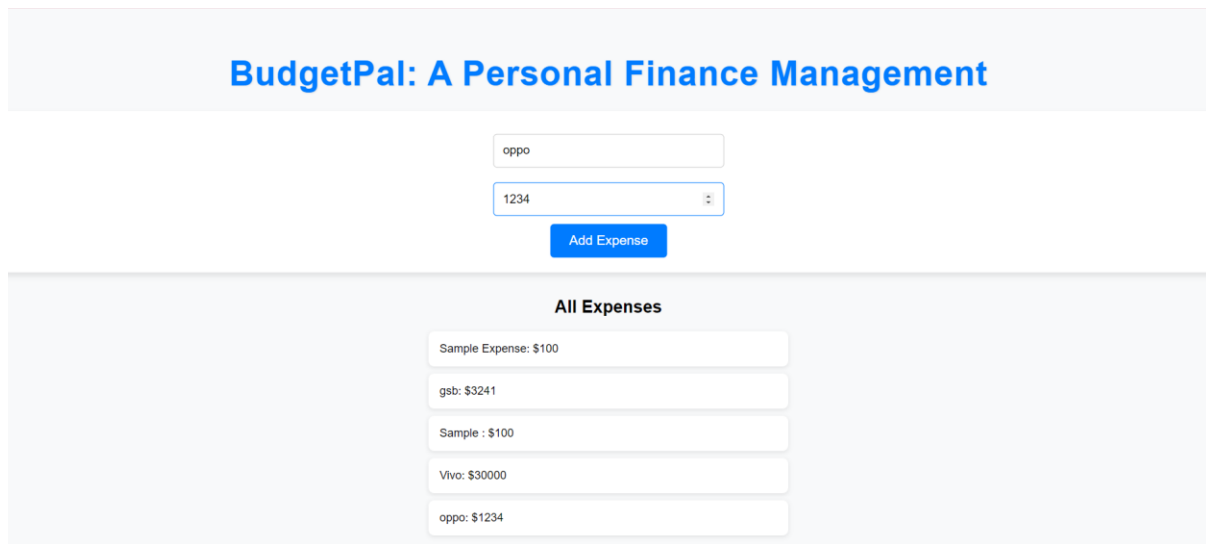
CHAPTER 4

CONCLUSION

In conclusion, **BudgetPal** demonstrates the effectiveness of the MERN stack in building a personal finance management application. By combining MongoDB for flexible data storage, Express.js and Node.js for efficient backend processing, and React.js for a dynamic frontend, the project showcases the power of modern web development in addressing real-world financial management needs. The application provides a comprehensive solution for users to track expenses, set budgets, and make informed financial decisions, with opportunities for further enhancements in future versions.

SCREENSHOTS

Fig – 1 Main Page



The screenshot displays the main interface of the BudgetPal application. At the top, a header reads "BudgetPal: A Personal Finance Management". Below this, there is a form for adding a new expense. The form consists of two input fields: the first is labeled "oppo" and the second is labeled "1234". A blue button labeled "Add Expense" is positioned below these fields. Underneath the form, a section titled "All Expenses" lists several sample entries, each in a separate box: "Sample Expense: \$100", "gsb: \$3241", "Sample : \$100", "Vivo: \$30000", and "oppo: \$1234".

Fig – 2 Database Stored

<pre>_id: ObjectId('671006de4ecad643fid1eb9f') name : "Sample " amount : 100 date : 2024-10-16T18:33:02.739+00:00 __v : 0</pre>
<pre>_id: ObjectId('6710a6738d77e390e614015b') name : "Vivo" amount : 30000 date : 2024-10-17T05:53:55.367+00:00 __v : 0</pre>
<pre>_id: ObjectId('6713a5a72d0e1619594fcae9') name : "oppo" amount : 1234 date : 2024-10-19T12:27:19.499+00:00 __v : 0</pre>

REFERENCES

- Kapoor, J., Dlabay, L. R., & Hughes, R. J. (2015). *Personal Finance*. McGraw-Hill Education.
- Crockford, D. (2008). *JavaScript: The Good Parts*. O'Reilly Media.
- McFarland, D. S. (2020). *HTML5: The Missing Manual* (3rd ed.). O'Reilly Media.