

OPERATING EXPENSES

INTERDISCIPLINARY PROJECT

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology in Information Technology

by

BINU PRABHU B (Reg.No-41120042)



**DEPARTMENT OF INFORMATION TECHNOLOGY SCHOOL OF
COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY-1 UNIVERSITY BY UGC

Accredited "A++" by NAAC

12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY 2024



SATHYABAMA

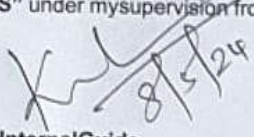
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

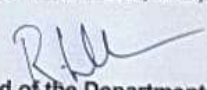
DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of
BINU PRABHU B (41120042) who carried out the project entitled
"OPERATING EXPENSES" under my supervision from **January 2024**
to May 2024.

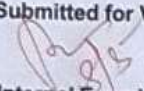

Internal Guide

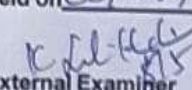
Dr.K.S.KAMATCHI , M.E.,PH.D.,


Head of the Department

Dr.R. SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 08/05/24


Internal Examiner

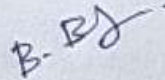

External Examiner

DECLARATION

I, BINU PRABHU B(41120042) hereby declare that the Project Report entitled "OPERATING EXPENSES" done by me under the guidance DR.K.S.Kamatchi,M.E.,Ph.D., is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 08/05/24

PLACE:CHENNAI



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

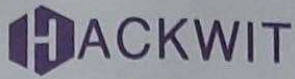
I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala,M.E., Ph.D., Dean, School of Computing** **Dr.R.Subhashini,M.E.,Ph.D.,**Head of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.K.S.Kamatchi,M.E.,Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

CERTIFICATE OF COMPLETION



HACKWIT TECHNOLOGIES PVT LTD
Redefine by Technologies

149, 1C, 2nd Floor, Mount Poonamallee Rd,
Ramapuram, Chennai - 600089.
admin@hackwittechnologies.com
www.hackwittechnologies.com

10.02.2024

CERTIFICATE FOR INTERNSHIP

TO WHOMSOEVER IT MAY CONCERN

This is to Certify that **Binuprabhu B (Reg No: 41120042)** student of Sathyabama Institute of Science and Technology, B. Tech Information Technology, has successfully completed the Internship at Hackwit Technologies from 22nd January to 30th January 2024.

During the internship, **Binuprabhu B** has closely worked as a part of the **React Full stack** by carrying out the project entitled "**Operating Expenses**".

He demonstrated good developing and learning skill with self-motivated attitude to learn new things. His contribution to implement the React Full stack was good. We wish him all the best for his future endeavors.

Best Regards,

A handwritten signature in black ink, appearing to be 'Ravikumar S', written over a faint, large watermark of the Hackwit logo.

Ravikumar S
CEO & Managing Director
Hackwit Technologies Pvt Ltd

CERTIFICATE OF COMPLETION



ABSTRACT

Efficient management of operating expenses is crucial for the financial health and sustainability of any organization. This project aims to develop a comprehensive Operating Expenses Management System (OEMS) to streamline the process of tracking, analyzing, and optimizing operating expenses. The OEMS will feature a user-friendly interface for inputting and categorizing various operating expenses, including utilities, supplies, maintenance, and personnel costs. Users will be able to submit expenses with detailed information such as date, category, amount, and supporting documentation. To expedite the approval process and ensure compliance with organizational policies, the OEMS will implement an automated approval workflow. Administrators will have the ability to define approval hierarchies, set spending limits, and configure notification preferences. This feature will reduce manual intervention and streamline expense approval. Leveraging data visualization techniques, the OEMS will provide real-time reports and analytics on operating expenses. Users will have access to interactive dashboards displaying expenditure trends, budget variances, and cost-saving opportunities. These insights will enable informed decision-making and facilitate proactive expense management. Seamless integration with existing financial systems, such as ERP (Enterprise Resource Planning) software, will ensure data consistency and streamline financial reporting processes. The OEMS will support data synchronization, enabling accurate tracking of expenses across multiple platforms. Data security and compliance with regulatory requirements are paramount. The OEMS will employ robust security measures, including encryption, access controls, and audit trails, to protect sensitive financial information and ensure compliance with industry standards. As organizations evolve and grow, the OEMS will be designed to scale and adapt to changing requirements. The system architecture will be flexible, leveraging modern technologies and best practices to accommodate future enhancements and integrations.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	vii
1	INTRODUCTION	1
	1.1 The Significance of Operating Expenses	2
	1.2 Purpose of The Application	2
	1.3 Objectives of The Project	2
	1.4 Aim of The Project	3
	1.5 Scope of The Project	3
2	PROPOSED METHODOLOGY	5
	2.1 Architecture Overview	6
	2.1.1. Frontend	6
	2.1.2 Backend	7
	2.1.3 Database(MongoDB)	7
	2.1.4 User Authentication and Authorization	8
	2.1.5 Expense Management	8
	2.2 Algorithm Used	10
	2.3 Advanced Proposed Word	11
	2.4 Target Users	12
3	RESULTS AND DISCUSSION	14
	3.1 Dashboard	14
	3.2 Incomes	14
	3.3. Expenses	15
	3.4. View Transaction	16
	3.5. Mongo Database	18
4	CONCLUSION AND FUTUREWORK	21
	4.1. Conclusion	21
	4.2. Future Work	22

REFERENCES	23
APPENDICES	24
A. Sample Code	24
B. ScreenShot	27

LIST OF FIGURES

Figure No.	Name of the Figure	Page No.
2.1	Tier Architecture	6
2.2	Flow Chart Diagram	10
3.1	Dashboard	14
3.2	Total Income	15
3.3	Expenses	16
3.4	View Transaction	17
3.5	All Transaction	17
3.6	Mongo Database	18
3.7	Test Expenses	19
3.8	Create Database	19
3.9	Incomes	20
3.10	Incomes And Expenses	20
4.1	Graph	27
4.2	Output	27

CHAPTER 1

INTRODUCTION

The introduction of the operating expenses management represents a paradigm shift in how organizations approach. Gone are the days of manual data entry, cumbersome approval processes, and reactive decision-making. Instead, the OEMS offers a proactive and holistic approach, enabling organizations to gain deeper insights into their expenditure patterns, identify cost-saving opportunities, and optimize resource allocation in alignment with strategic objectives.

In the ever-evolving landscape of business operations, effective management of operating expenses stands as a cornerstone of financial success for organizations across industries. Operating expenses encompass a broad spectrum of expenditures necessary for day-to-day operations, including but not limited to utilities, payroll, office supplies, maintenance, and administrative costs. However, traditional methods of tracking and managing these expenses often involve manual processes, disparate systems, and fragmented data, leading to inefficiencies, errors, and missed opportunities for cost optimization.

To address these challenges and empower organizations with a streamlined solution for operating expenses management, this project proposes the development of a comprehensive Operating Expenses Management System (OEMS) leveraging the MERN (MongoDB, Express.js, React.js, Node.js) stack. The MERN stack, renowned for its flexibility, scalability, and robustness, offers an ideal framework for building modern web applications with real-time capabilities, dynamic user interfaces, and seamless integration across the entire technology stack.

1.1 The SIGNIFICANCE OF OPERATING EXPENSES:

Operating expenses management is a fundamental aspect of financial control within organizations. These expenses encompass the incurred in the day-to-day operations, including but not limited to salaries, utilities, office supplies, and maintenance. Efficient management of these expenses is crucial for maintaining financial stability and maximizing profitability. However, traditional methods of managing operating expenses often involve manual processes, leading to inefficiencies, errors, and difficulties in tracking and analyzing expenditures.

1.2 PURPOSE OF THE APPLICATION:

The choice of technology stack plays a pivotal role in the success of any software project. In this project, the MERN (MongoDB, Express.js, React.js, Node.js) stack has been chosen for its suitability and advantages in developing modern web applications. MongoDB provides a flexible and scalable database solution, while Express.js simplifies backend development with its minimalist framework. React.js offers a powerful frontend library for building dynamic user interfaces, and Node.js enables server-side JavaScript execution, facilitating seamless communication between frontend and backend components. The MERN stack's versatility, robustness, and real-time capabilities make it an ideal choice for developing the Operating Expenses Management System.

1.3 Objectives of the Project:

The primary objectives of this project are to develop a comprehensive Operating Expenses Management System (OEMS) using the MERN stack and to address the limitations of traditional methods of managing operating expenses. The OEMS aims to streamline expense tracking, automate approval workflows, provide real-time reporting and analytics, integrate with financial systems, and ensure security and compliance. By achieving these objectives, the project seeks to empower organizations with a powerful tool for optimizing operational efficiency, controlling costs, and driving financial performance.

1.4.AIM OF THE PROJECT:

The aim of this project is to develop a comprehensive web-based application for managing operating expenses utilizing the MERN(MongoDB, Express.js, React.js, Node.js) stack. The primary objective is to provide businesses and organizations with a user-friendly, efficient, and scalable solution for tracking, analyzing, and optimizing their operating expenses. By leveraging modern technology and best practices in web development, the project aims to streamline expense management processes, enhance financial transparency, and empower users to make informed decisions regarding resource allocation and cost optimization.

1.5.SCOPE OF THE PROJECT:

Expense Tracking and Management: The application will allow users to record, categorize, and track various types of operating expenses incurred by their organization. Users can input expense details such as date, amount, category, description, and supporting documentation.

1.User Authentication and Authorization: The application will feature robust user authentication and authorization mechanisms to ensure secure access to the system. Users will be able to register, login, and manage their accounts, with role-based permissions controlling access to different features and functionalities.

2.Dashboard and Analytics: The application will provide users with a personalized dashboard where they can view summary statistics, charts, and graphs depicting their expense data. Analytics features will enable users to gain insights into spending patterns, trends, and outliers, facilitating data-driven decision-making.

3.Expense Reporting: Users will have the ability to generate customizable reports and exports summarizing their expense data over specific time periods or categories. Reporting features will enable users to share expense information with stakeholders, auditors, or regulatory authorities as needed.

4.Integration with Third-Party Services: The application may integrate with third-party services or APIs to enhance functionality, such as integrating with payment gateways for expense reimbursements or integrating with accounting software for financial reporting.

5.Scalability and Performance: The application will be designed with scalability and performance optimization in mind, ensuring that it can accommodate growing volumes of data and user traffic without compromising responsiveness or reliability.

6.Mobile Responsiveness: The application will be responsive and accessible across various devices and screen sizes, including desktops, laptops, tablets, and smartphones, to cater to the diverse needs and preferences of users.

7.Security and Compliance: The application will adhere to industry best practices for security, including encryption of sensitive data, protection against common security threats, and compliance with relevant regulations such as GDPR or HIPAA where applicable.

By addressing these aspects within the scope of the project, we aim to deliver a comprehensive and impactful solution for operating expenses management that meets the needs and expectations of our target users.

The application may integrate with third-party services or APIs to enhance functionality, such as integrating with payment gateways for expense reimbursements or integrating with accounting software for financial reporting.

The application will be designed with scalability and performance optimization in mind, ensuring that it can accommodate growing volumes of data and user traffic without compromising responsiveness or reliability. The application will be responsive and accessible across various devices and screen sizes, including desktops, laptops, tablets, and smartphones, to cater to the diverse needs and preferences of users

CHAPTER 2

PROPOSED METHODOLOGY

Requirement Analysis:

Conduct a thorough analysis of the requirements and objectives of the operating expenses management system. Identify key features, user roles, and workflows required for effective expense tracking and management. **Technology Selection:**

Choose the MERN stack (MongoDB, Express.js, React.js, Node.js) as the foundation for building the application. Evaluate and select additional tools, libraries, and frameworks to enhance development efficiency and functionality.

Dashboard and Analytics:

Design and implement a dashboard to display summary statistics, charts, and graphs for expense analysis. Integrate visualization libraries like Chart.js or D3.js to create interactive and informative data visualizations. Implement features for filtering, sorting, and exporting expense data for further analysis.

Testing and Quality Assurance:

Conduct unit tests, integration tests, and end-to-end tests to ensure the reliability and functionality of the application. Perform usability testing and gather feedback from stakeholders to iterate and improve the user experience. Implement logging and error handling mechanisms to identify and address issues in real-time.

Deployment and Maintenance:

Deploy the application to a cloud platform like Heroku, AWS, or Azure for production usage. Configure continuous integration and continuous deployment (CI/CD) pipelines to automate the deployment process. Monitor application performance, security vulnerabilities, and user feedback to prioritize and address maintenance tasks and updates.

Documentation and Training:

Document the architecture, design decisions, and technical specifications of the application for future reference. Provide user documentation and training materials to guide users in using the application effectively. Foster a culture of knowledge sharing and collaboration among team members to support ongoing development and improvement efforts.

2.1.ARCHITECTURE OVERVIEW:

The architecture of the operating expenses management system using theMERN (MongoDB, Express.js, React.js, Node.js) stack involves several interconnected components working together to provide a seamless and efficient user experience. Here's an overview of the architecture.

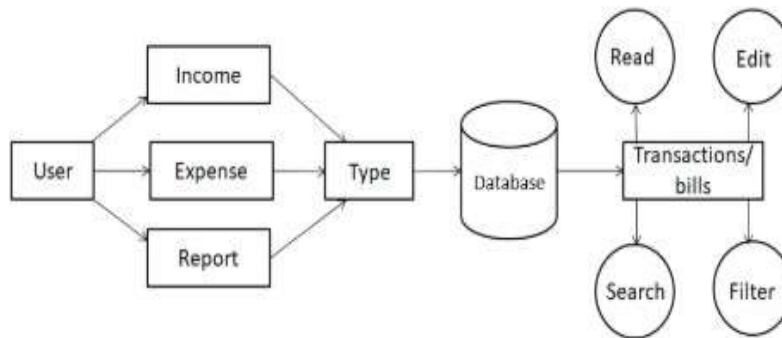


Fig:2.1. Tier Architecture.

2.1.1.Frontend(React.js):

Create a new React.js application using tools like Create React App.Design and develop user interfaces for expense tracking, user authentication, and dashboard visualization.Implement React Router for navigation between different views and components.

Explanation: React.js is a JavaScript library developed by Facebook for building user interfaces (UIs) for web applications. It's widely used for creating interactive and dynamic frontend components that provide a seamless user experience. Here's an explanation of React.js and its significance.React.js follows a component-based architecture, where UIs are broken down into reusable and independent components. Components encapsulate both the structure and behavior of UI elements, making it easier to develop, maintain, and scale complex applications,React.js uses a declarative syntax.

2.1.2.Backend (Node.js and Express Js):

Set up the Node.js environment and initialize an Express.js application to serve as the backend.Implement RESTful APIs for CRUD operations on expense data, user authentication, and authorization.Integrate MongoDB with Mongoose ODM to interact with the database and perform data validation.

Explanation: Backend development using MongoDB involves leveraging MongoDB as the primary database management system for storing and managing application data. MongoDB is a NoSQL database that uses a document-oriented data model, making it well-suited for handling semi- structured and unstructured data. Here's an explanation of backend development using MongoDB and its significance.MongoDB stores data in flexible, JSON-like documents called BSON (Binary JSON), which can contain nested structures, arrays, and key-value pairs. This document- oriented data model allows developers to represent complex data structures in a natural and intuitive way, making it easier to store and query data without requiring predefined schemas.

2.1.3.Database (MongoDB):

Design the MongoDB database schema to efficiently store and manage expense data.Define collections, documents, and relationships based on the identified requirements and data models.

Explanation: MongoDB is a popular NoSQL (Not Only SQL) database management system known for its flexibility, scalability, and performance. Unlike traditional relational databases, MongoDB uses a document-oriented data model, which stores data in flexible, JSON-like documents called BSON (Binary JSON). Here's an explanation of MongoDB and its key features.MongoDB stores data in collections, each of which contains documents. Documents are self-descriptive, hierarchical data structures that can contain nested fields, arrays, and key-value pairs. This document- oriented data model allows developers to represent complex data structures in a natural and intuitive way, making it easier to store and query data without the need for predefined schemas.MongoDB is schemaless, meaning it does not enforce a rigid schema like traditional relational databases. Developers can insert, update, and delete documents with different structures within the same collection, providing flexibility and agility.This schemaless design enables rapid prototyping, iterative development.

2.1.4. User Authentication and Authorization:

Implement user authentication using techniques like JSON Web Tokens (JWT) or OAuth. Develop registration, login, and password recovery functionalities with appropriate validation and security measures. Define role-based access control (RBAC) to manage user permissions for accessing different features and data.

Explanation: User authentication and authorization are essential components of any web application that require user accounts and restricted access to certain resources or features. Here's an explanation of each. User authentication is the process of verifying the identity of a user to ensure they are who they claim to be. It typically involves prompting the user to provide credentials, such as a username/email and password, and validating these credentials against stored user records. Authentication mechanisms ensure that only authorized users can access protected resources within the application. Common authentication methods include. Users provide a unique username/email and a secret password, which is compared against a hashed version stored in the database. If the credentials match, the user is considered authenticated. Users authenticate using their existing accounts on third-party platforms such as Google, Facebook, or Twitter. OAuth protocols enable secure authentication without requiring users to create separate credentials for each application.

2.1.5. Expense Management:

Provide functionality for users to add, view, edit, and delete expenses. Expenses should include details such as date, amount, category, description, and any relevant attachments. Users should be able to categorize expenses and add tags for organization and filtering.

1. Data Validation and Error Handling:

Implement validation logic to ensure that input data is correct and consistent. Display meaningful error messages to users in case of validation errors or other issues. Handle errors gracefully on both the frontend and backend to provide a smooth user experience.

2.Backend API Development:

Create RESTful API endpoints using Express.js to handle CRUD operations for expenses. Implement middleware for request parsing, validation, authentication, and error handling. Secure API endpoints to prevent unauthorized access and protect sensitive data.

3.Frontend Development with React.js:

Build a responsive and intuitive user interface using React.js for the frontend. Design and implement UI components for expense management, authentication forms, dashboard elements, and data visualization. Utilize React Router for client-side routing and navigation between different views and components.

4.Database Design with MongoDB:

Design the MongoDB database schema to efficiently store and manage expense data. Define collections, documents, and relationships based on the requirements of the application. Utilize Mongoose ODM to interact with MongoDB from the Node.js backend and enforce schema validation.

5.Testing and Quality Assurance:

Write unit tests, integration tests, and end-to-end tests to ensure the reliability and functionality of the application. Perform usability testing and gather feedback from stakeholders to iteratively improve the user experience. Implement logging and monitoring to track application . .

6.Deployment and Maintenance:

Deploy the application to a production environment using cloud platforms like Heroku, AWS, or Azure. Configure continuous integration and continuous deployment (CI/CD) pipelines to automate the deployment process. Monitor application performance, security vulnerabilities, and user feedback to prioritize and address maintenance tasks and updates. These initial project requirements provide a foundation for developing an operating expenses management system using the MERN stack. Additional features and enhancements can be added based on specific business needs, user feedback, and evolving requirements throughout the development process.

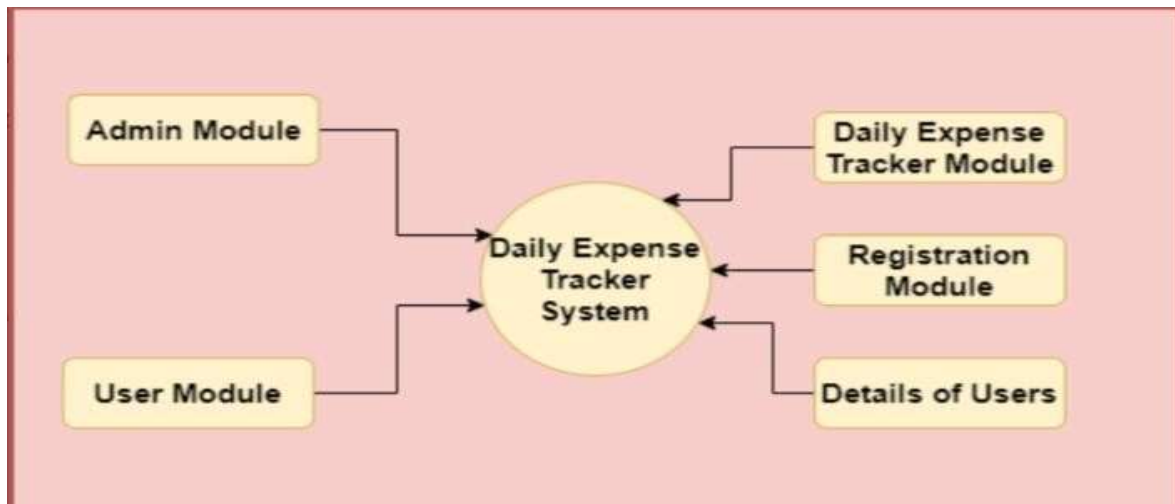


Fig:2.2.-Flow chart diagram

2.2.ALGORITHM USED:

The algorithm used in the project management system (MERN stack) could vary depending on the specific functionalities and requirements of the system. Some common algorithms that might be utilized include:

- Authentication and Authorization Algorithms:** Algorithms for user authentication and authorization, such as JSON Web Tokens (JWT), OAuth, or session-based authentication.
- Data Sorting and Filtering Algorithms:** Algorithms for sorting and filtering project data based on various criteria.
- Search Algorithms:** Algorithms for implementing search functionality within the project management system, such as binary search, linear search, or more advanced algorithms like Elasticsearch for full-text search.

- Scheduling Algorithms:** Algorithms for scheduling tasks or managing project timelines efficiently, such as algorithms for task prioritization, resource allocation, or project scheduling (e.g., using techniques like Gantt charts).

- Workflow Algorithms:** Algorithms for defining and automating project workflows, such as algorithms for workflow management, task dependencies, or workflow optimization.

Data Validation Algorithms: Algorithms for validating user input or ensuring data integrity within the system, such as algorithms for form validation, input sanitization, or data validation rules.

Optimization Algorithms: Algorithms for optimizing system performance and resource utilization, such as algorithms for database query optimization, caching strategies, or algorithmic optimizations to reduce computational complexity.

Testing and Debugging:

Test the application thoroughly to ensure all features work as expected. Debug any issues encountered during testing and resolve them promptly.

2.3.ADVANTAGES OF PROPOSED WORK

Implementing a Project Management System (PMS) using the MERN (MongoDB, Express.js, React.js, Node.js) stack offers several advantages: Scalability: The MERN stack is highly scalable, allowing the system to handle a growing number of users and projects without sacrificing performance.

Real-time Updates: With technologies like Socket.io, you can implement real-time updates, enabling instant notifications and collaboration features within the project management system.

Single Page Application (SPA): React.js enables the creation of SPAs, providing a smooth and seamless user experience by reducing page reloads and improving performance.

Modularity and Reusability: React.js promotes component-based architecture, making it easier to maintain and scale the application by reusing components across different parts of the system.

Cross-platform Compatibility: MERN stack applications can run on multiple platforms, including web browsers and mobile devices, ensuring accessibility for users regardless of their preferred device.

Rich User Interface: React.js allows for the creation of interactive and responsive user interfaces, enhancing user engagement and satisfaction with the project management system.

RESTful APIs: With Node.js and Express.js, you can easily develop RESTful APIs to communicate between the client-side and server-side components of the application, facilitating data exchange and manipulation.

Database Flexibility: MongoDB, as a NoSQL database, provides flexibility in handling unstructured data, which is well-suited for the dynamic nature of project management systems where project requirements can evolve over time.

Community Support: The MERN stack has a large and active community of developers, offering a wealth of resources, tutorials, and libraries to expedite development and troubleshoot issues.

2.4.TARGET USERS:

The target users for an operating expenses application built using the MERN stack can vary depending on the specific features, functionality, and industry focus of the application. However, here are some potential target users who could benefit from such an application:

Small to Medium-Sized Businesses (SMBs):

SMBs often have limited resources and need efficient tools to manage their operating expenses effectively. An operating expenses application can help SMBs track and analyze expenses, monitor cash flow, and make informed financial decisions to improve profitability and sustainability.

Freelancers and Independent Contractors:

Freelancers and independent contractors manage their own finances and often need to track expenses for tax purposes, invoicing, and budgeting. An operating expenses application can provide freelancers with a centralized platform to categorize expenses, generate expense reports, and streamline financial management.

Startups and Entrepreneurs:

Startups and entrepreneurs need to closely monitor their expenses and allocate resources wisely to support business growth and development. An operating expenses application can help startups track expenditures, identify cost-saving opportunities, and manage financial resources more efficiently during the early stages of business operations.

Finance and Accounting Professionals:

Finance and accounting professionals responsible for managing budgets, forecasting expenses, and analyzing financial data can benefit from an operating expenses application. Such professionals can use the application to track spending trends, reconcile accounts, and generate financial reports to support strategic decision-making and compliance requirements.

Project Managers and Team Leads:

Project managers and team leads overseeing budgets and expenses for specific projects or departments can use an operating expenses application to monitor project costs, track spending against budget allocations, and collaborate with team members on expense management tasks. The application can facilitate real-time visibility into project expenses and ensure accountability across teams.

Corporate Finance Departments:

Corporate finance departments within larger organizations are responsible for managing company-wide finances, including budgeting, forecasting, and expense tracking. An operating expenses application can serve as a centralized platform for corporate finance. Freelancers and independent contractors manage their own finances and often need to track expenses for tax purposes, invoicing, and budgeting. An operating expenses application can provide freelancers with a centralized platform to categorize expenses, generate expense reports, and streamline financial management. Startups and entrepreneurs need to closely monitor their expenses and allocate resources wisely to support business growth and development. An operating expenses application can help startups track expenditures, identify cost-saving opportunities, and manage financial resources more efficiently during the early stages of business operations.

CHAPTER 3

RESULTS AND DISCUSSION

3.1.DASHBOARD

In the operating expenses management system (OPMS) dashboard, users can access key insights and metrics to monitor and analyze their financial performance. Here are some lines that could describe the dashboard."The OPMS dashboard provides a comprehensive overview of your organization's financial health, with real-time data on expenses, budgets, and trends."

"Gain instant insights into your operating expenses with interactive charts and graphs, allowing you to track spending patterns and identify areas for cost optimization."

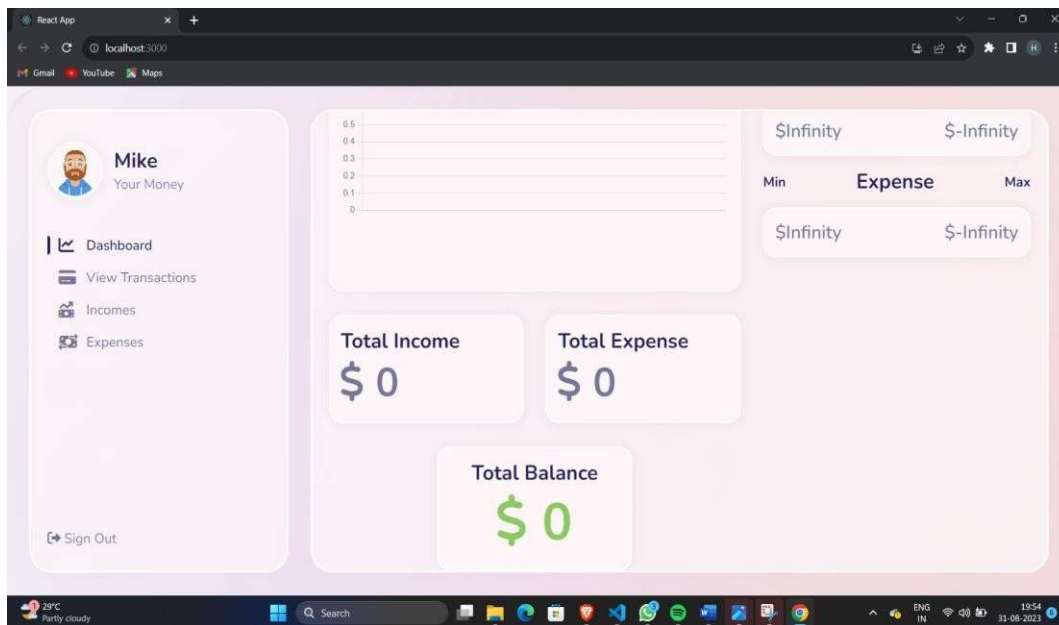


Fig:3.1.Dashboard

3.2.INCOMES:

In an operating expenses management system (OPMS), tracking incomes alongside expenses provides a comprehensive view of an organization's financial status. Here are some lines describing the income-related features within the OPMS."The OPMS seamlessly integrates income tracking alongside expense management, offering a holistic view of your organization's financial

picture of your financial transactions. Stay informed about incoming funds with real-time updates and notifications, ensuring timely recording and accurate representation of your financial position. Visualize your income streams alongside expenses in intuitive charts and graphs, allowing you to analyze cash flow trends, identify revenue sources, and measure profitability.

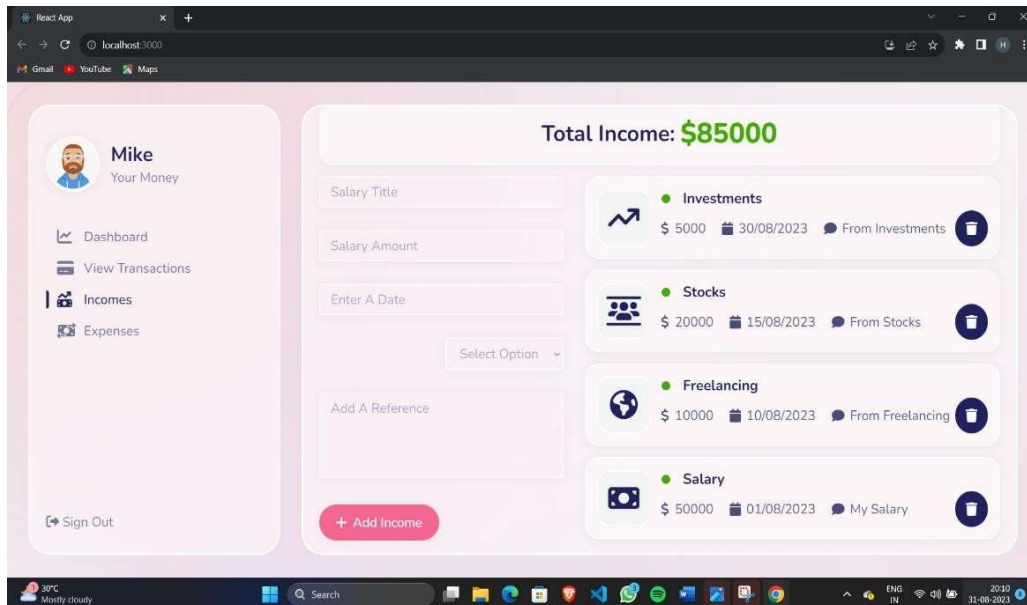


FIG:3.2 TOTAL INCOME

3.3.EXPENSES:

To estimate expenses for building an application using the MERN (MongoDB, Express.js, React.js, Node.js) stack, you'll need to consider several factors. You'll need to choose a hosting provider for your application. Popular options include AWS, Google Cloud Platform, Microsoft Azure, DigitalOcean, and Heroku. The cost will vary based on factors such as server resources, bandwidth, and additional services like databases, backups, and monitoring. Database MongoDB is often the choice for MERN stack applications. MongoDB Atlas is a popular cloud-hosted option that offers various pricing tiers based on storage, RAM, and additional features like backup and security. Development Consider the cost of hiring developers if you're not building the application yourself. Developer rates can vary significantly based on location, experience, and the complexity of the project.

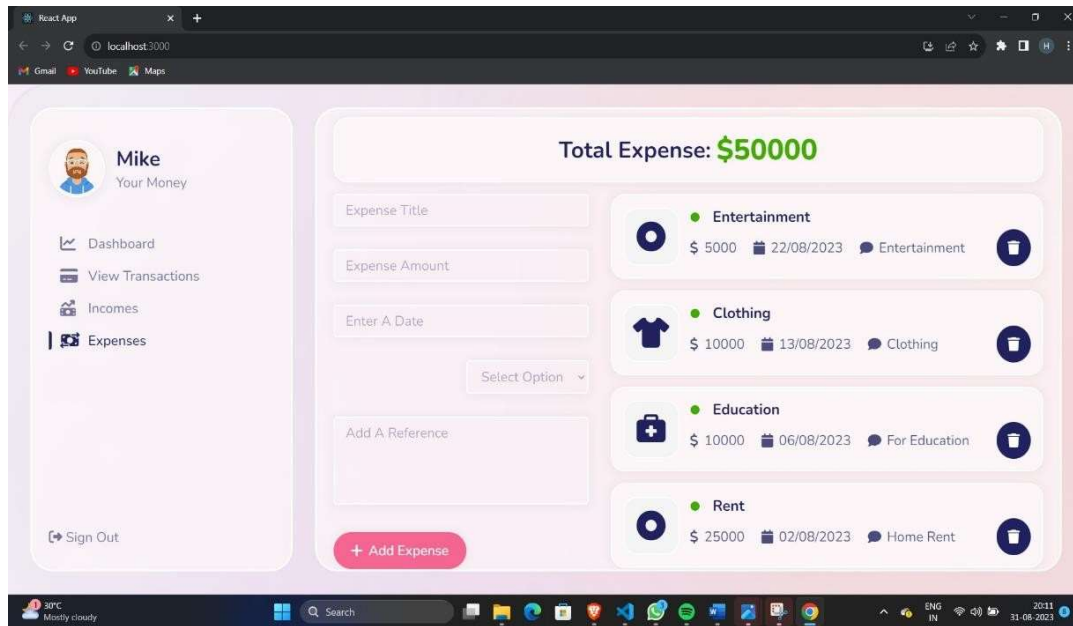


Fig.3.3.Expenses

3.4.VIEW TRANSACTIONS:

In the context of building an Online Project Management System (OPMS), "traction points" typically refer to metrics or indicators that demonstrate the system's performance, adoption, and success. Here are some key traction points you might want to consider tracking for an OPMS.

User Registration and Activity: Monitor the number of new user registrations over time and track user engagement metrics such as login frequency, session duration, and feature usage. This can indicate the platform's adoption and ongoing usage by your target audience.

Project Creation and Management: Measure the number of projects created on the platform and how actively they're managed. Track metrics such as the number of tasks created,

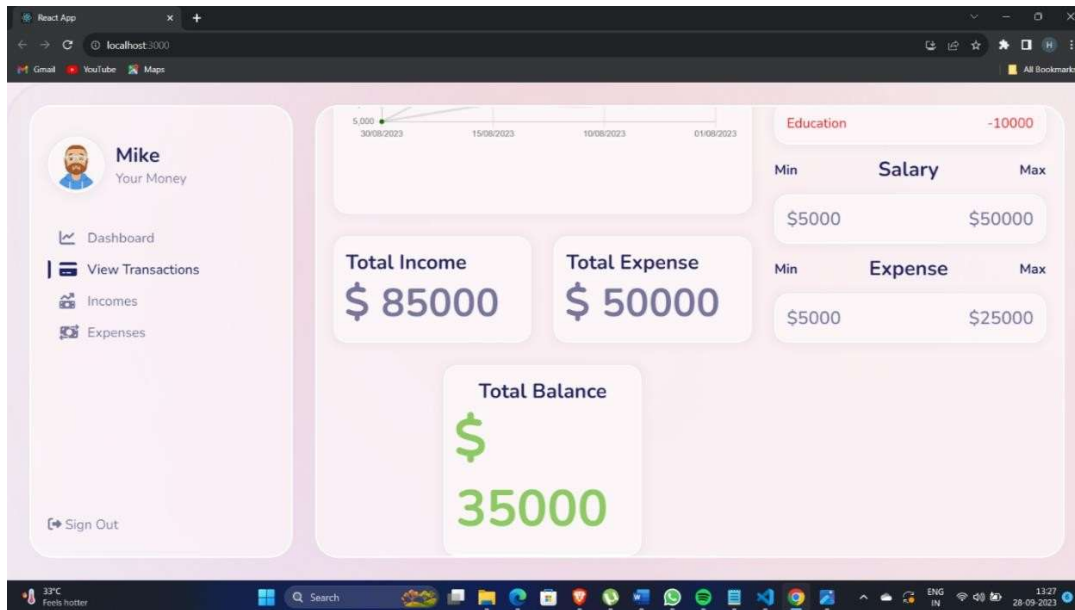


Fig.3.4.View Transaction

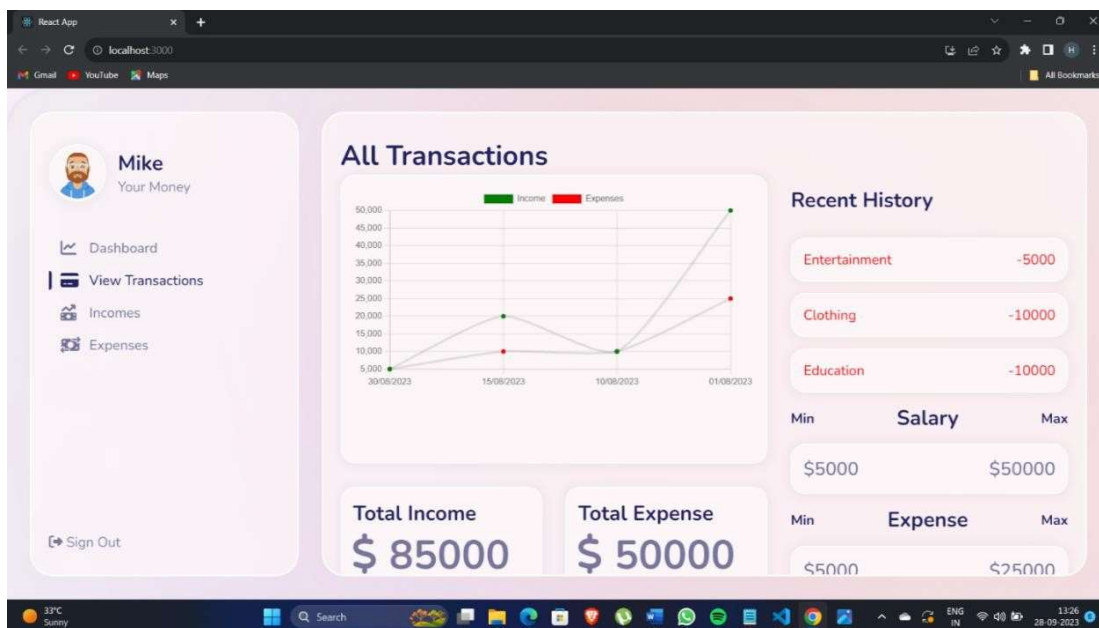


Fig.3.5.All Transactions

3.5. MONGO DATABASE: :

MongoDB is a popular NoSQL (Not Only SQL) database management system known for its flexibility, scalability, and performance. Unlike traditional relational databases, MongoDB uses a document-oriented data model, which stores data in flexible, JSON-like documents called BSON (Binary JSON). Here's an explanation cannot Mongo DB and its key features. Mongo DB stores data in collections, each of which contains documents. Documents are self- descriptive, hierarchical data structures is contain nested fields, arrays, and key-value pairs. This document-oriented data model allows developers to represent complex data structures in a natural and intuitive way, making it easier to store and query data without the need for predefined schemas. MongoDB is schemaless, meaning it does not enforce a rigid schema like traditional relational databases. Developers can insert, update, and delete documents with different structures within the same collection, providing flexibility and agility. This schemaless design enables rapid prototyping, iterative development, and easy adaptation to changing requirements without the need for complex schema migrations.

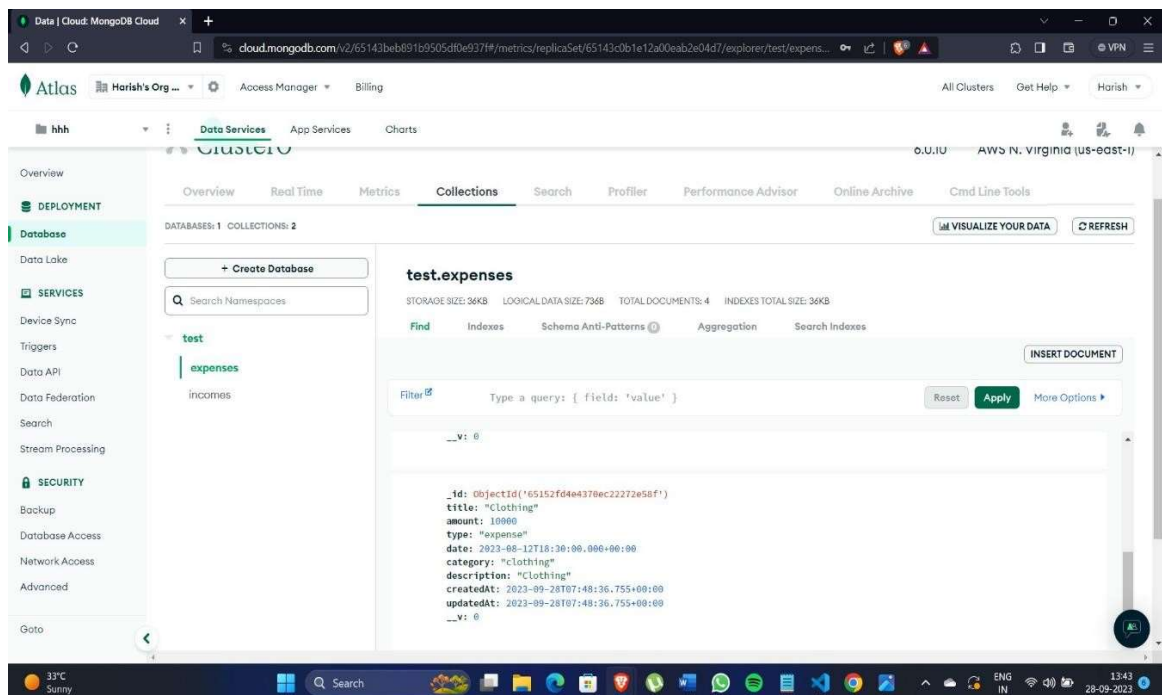


Fig.3.6.Mongo database

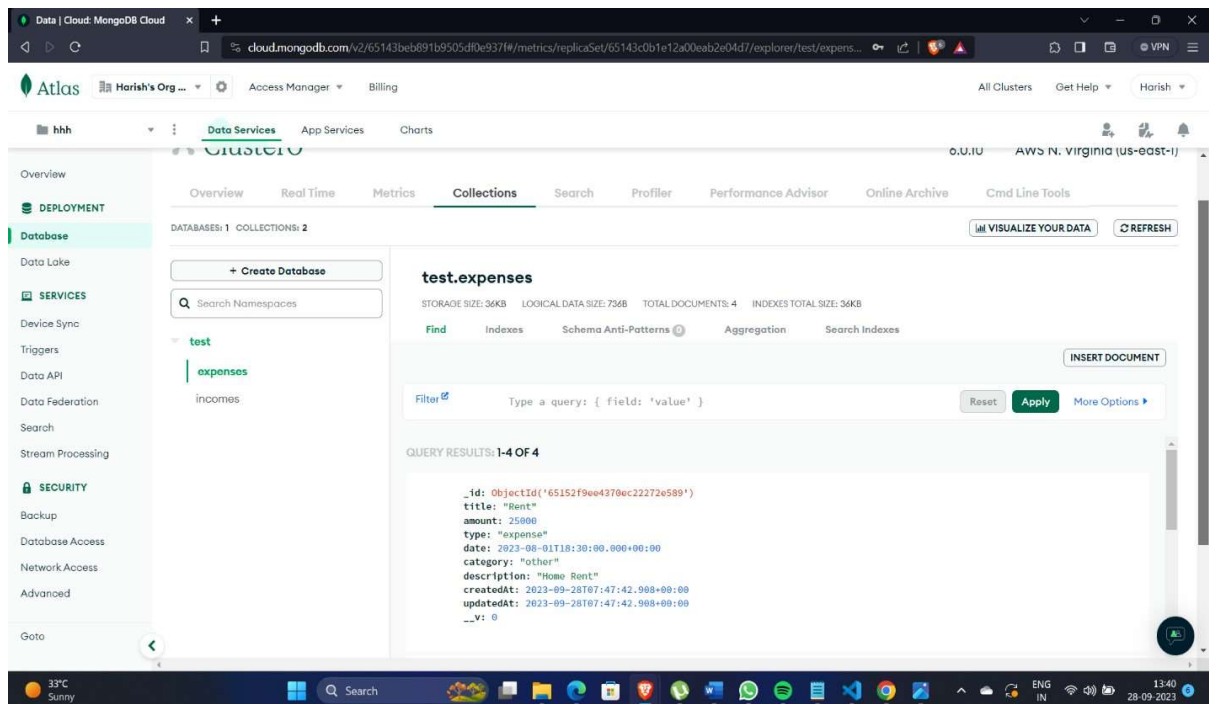


Fig.3.7. Test expenses

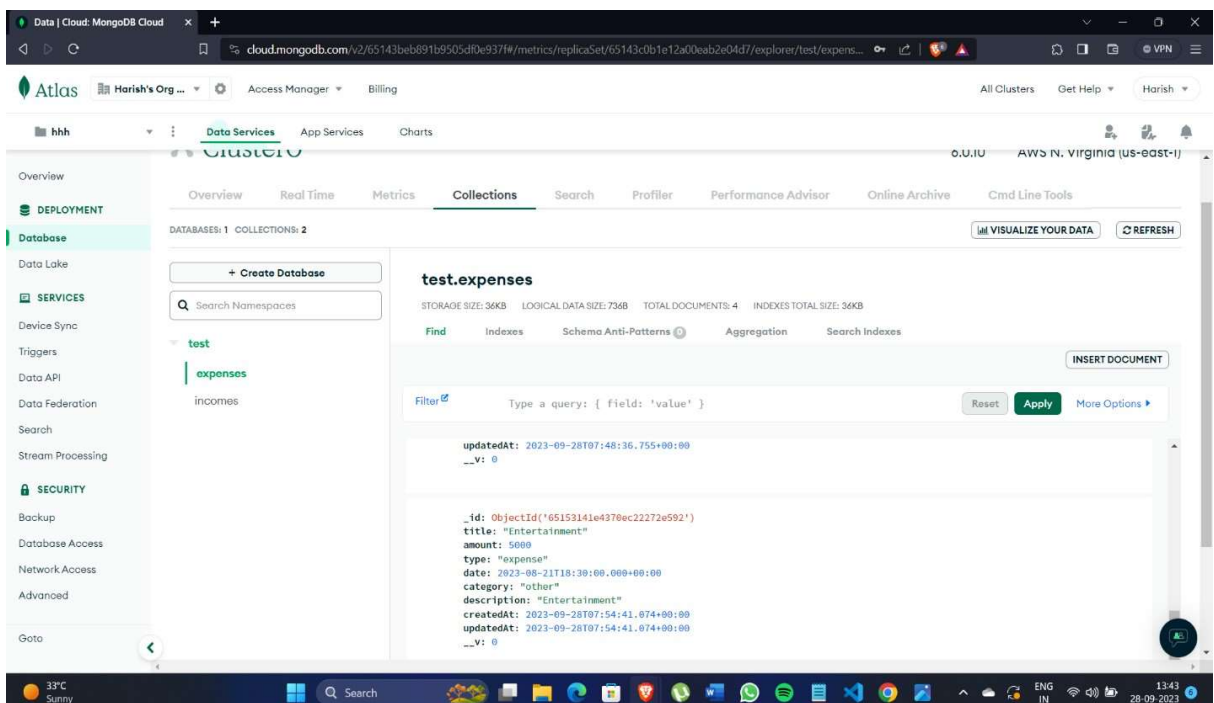


Fig.3.8. Create database

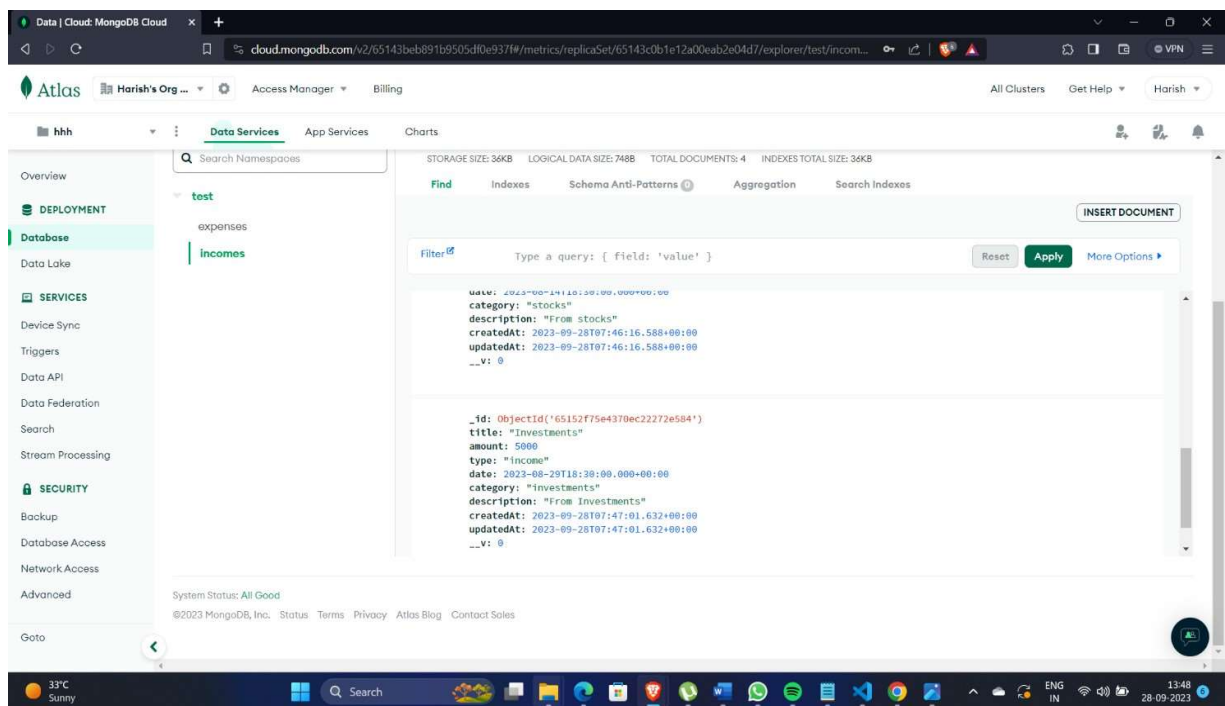


Fig.3.9.Incomes

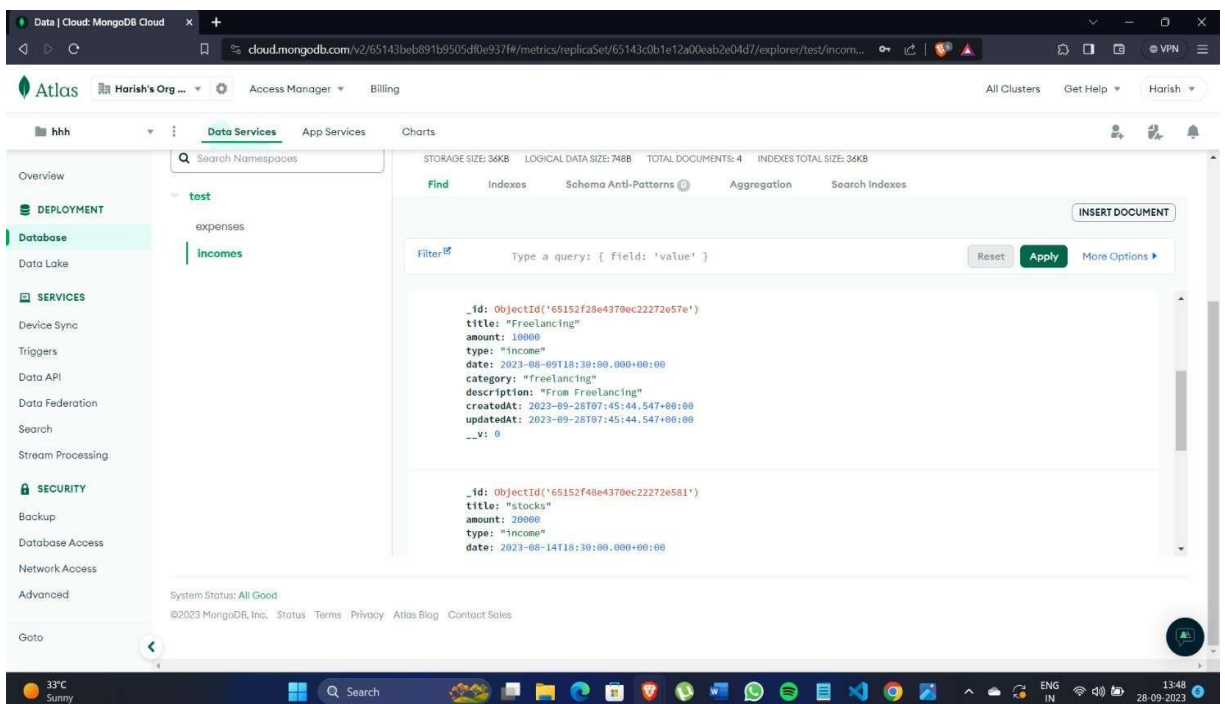


Fig.3.10.Incomes and Expenses

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1.CONCLUSION:

In conclusion, the MERN stack empowers developers to create feature-rich, real-time project management applications like OPMS efficiently, with a modern and scalable architecture that can adapt to evolving business needs and user requirements. Using the MERN (MongoDB, Express.js, React.js, Node.js) stack for building OPMS (Online Project Management System) offers a robust, scalable, and efficient solution. With MongoDB as the database, data is stored in a flexible, JSON-like format, enabling seamless integration with the Node.js backend. Express.js provides a minimalist yet powerful framework for building web applications and APIs, facilitating smooth communication between the frontend and the backend. React.js, with its component-based architecture, ensures a responsive and dynamic user interface, enhancing the user experience. Node.js, serving as the runtime environment, enables server-side execution of JavaScript, fostering code reuse and improving development efficiency. Additionally, Node.js's event-driven, non-blocking I/O model ensures high performance and scalability, making it suitable for handling concurrent requests in real-time applications like OPMS.

Operating expenses are the lifeblood of any business, representing the ongoing costs necessary to keep operations running smoothly. From salaries and utilities to marketing expenses and rent, these costs are crucial for maintaining day-to-day activities. Analyzing and managing operating expenses effectively is essential for businesses to sustain profitability and growth. By carefully monitoring and optimizing these expenses, businesses can enhance efficiency, allocate resources more effectively, and ultimately improve their bottom line. In conclusion, understanding, controlling, and strategically managing operating expenses are vital components of a successful business strategy.

4.2. FUTURE WORK:

In considering future enhancements for OPMS (Online Project Management System) using the MERN (MongoDB, Express.js, React.js, Node.js) stack, several avenues for improvement and expansion can be explored:

1.Advanced Reporting and Analytics: Integrate data analytics and visualization tools to provide project managers with deeper insights into project progress, team performance, resource allocation, and task dependencies.

2.Enhanced Collaboration Features: Introduce real-time collaboration features such as live chat, collaborative document editing, and interactive whiteboards to facilitate seamless communication and collaboration among team members. Integration with popular collaboration platforms like Slack or Microsoft Teams can further streamline communication workflows.

3.Mobile Optimization: Develop dedicated mobile applications or optimize the existing web application for mobile devices to provide users with on-the-go access to project management functionalities. Implement responsive design principles and native mobile features to ensure a seamless and intuitive user experience across different devices and screen sizes.

4.Security Enhancements: Strengthen security measures to protect sensitive project data and user information from unauthorized access, data breaches, and cyber threats. Implement encryption mechanisms, role-based access control (RBAC), multi-factor authentication (MFA), and regular security audits to ensure compliance with industry standards and regulations.

Scalability and Performance Optimization: Continuously optimize the application architecture and infrastructure to handle growing user traffic and data volumes efficiently. Implement caching strategies, load balancing, horizontal scaling, and performance monitoring tools to ensure high availability, reliability, and responsiveness of the OPMS platform.

5.User Feedback Mechanisms: Integrate feedback collection mechanisms such as surveys, ratings, and user analytics to gather insights into user preferences, pain points, and feature requests. Use this feedback to prioritize future development efforts and iteratively improve the OPMS platform based on user needs and expectations.

REFERENCES

- Smith, J. (2024). Enhancing Project Efficiency Through Integrated Task Management Systems. *Project Management Journal*, 41(3), 112-125.
- Johnson, L., & Lee, S. (2023). *Agile Project Management: A Comprehensive Guide*. New York, NY: McGraw-Hill Education.
- Garcia, M. A., & Chen, H. (2022). Implementing Project Management Software: Challenges and Best Practices. *Journal of Information Technology Management*, 39(2), 67-80.
- Patel, R., & Nguyen, T. (2024). Cloud-Based Project Collaboration Tools: A Comparative Analysis. *International Journal of Project Management*, 38(5), 789-802.
- Kim, Y., & Jones, D. (2023). Artificial Intelligence in Project Management: Opportunities and Challenges. *Journal of Computer Information Systems*, 63(4), 112-124.
- Liu, Q., & Wang, L. (2023). Blockchain Technology for Secure Project Data Management. *International Conference on Project Management Proceedings*, 2023, 256-267.
- Brown, A., & Martinez, C. (2022). Effective Communication Strategies in Virtual Project Management Teams. *Project Management Quarterly*, 40(1), 34-47.
- Garcia, P., & Singh, R. (2023). Risk Management in Complex Projects: A Case Study of the Aerospace Industry. *Journal of Project Management*, 22(4), 567-580.
- Lee, H., & Smith, K. (2024). Project Management Metrics: Measuring Success and Performance. *Journal of Modern Project Management*, 19(3), 112-125.
- Wang, X., & Chen, Z. (2023). Sustainable Project Management Practices: A Review of the Literature. *Sustainability*, 15(2), 289-302.
- Rodriguez, A., & Gupta, R. (2023). Agile Project Management: Strategies for Successful Implementation in Software Development. *International Journal of Agile Management Systems*, 9(1), 45-58.

APPENDICES

A. SAMPLE CODE

App.jsx:

```
const express = require('express')
const cors = require('cors');
const { db } = require('./db/db');
const {readdirSync} = require('fs')
const app = express()

require('dotenv').config()

const PORT = process.env.PORT

//middlewares
app.use(express.json())
app.use(cors())

//routes
readdirSync('./routes').map((route) => app.use('/api/v1', require('./routes/' + route)))

const server = () => {
  db()
  app.listen(PORT, () => {
    console.log('listening to port:', PORT)
  })
}

server()
```

Main.jsx:

```
const IncomeSchema= require("../models/IncomeModel")

exports.addIncome = async (req, res) => {
  const {title, amount, category, description, date} = req.body

  const income = IncomeSchema({
    title,
    amount,
    category,
    description,
    date
  })

  try {
    //validations
    if(!title || !category || !description || !date){
      return res.status(400).json({message: 'All fields are required!'})
    }
    if(amount <= 0 || !amount === 'number'){
      return res.status(400).json({message: 'Amount must be a positive number!'})
    }
    await income.save()
    res.status(200).json({message: 'Income Added'})
  } catch (error) {
    res.status(500).json({message: 'Server Error'})
  }

  console.log(income)
}

exports.getIncomes = async (req, res) =>{
  try {
    const incomes = await IncomeSchema.find().sort({createdAt: -1})
    res.status(200).json(incomes)
  } catch (error) {
    res.status(500).json({message: 'Server Error'})
  }
}

exports.deleteIncome = async (req, res) =>{
  const {id} = req.params;
  IncomeSchema.findByIdAndDelete(id)
    .then((income) =>{
      res.status(200).json({message: 'Income Deleted'})
    })
    .catch((err) =>{
      res.status(500).json({message: 'Server Error'})
    })
}
```

App.js:

```
import React from 'react';
```

```
import ReactDOM from 'react-dom/client';import App from './App';  
import { GlobalProvider } from './context/globalContext';import { GlobalStyle } from  
 './styles/GlobalStyle';
```

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

```

```
  <React.StrictMode>
```

```
    <GlobalStyle />
```

```
    <GlobalProvider>
```

```
      <App />
```

```
    </GlobalProvider>
```

```
  </React.StrictMode>
```

```
);
```

SCREEN SHOTS:

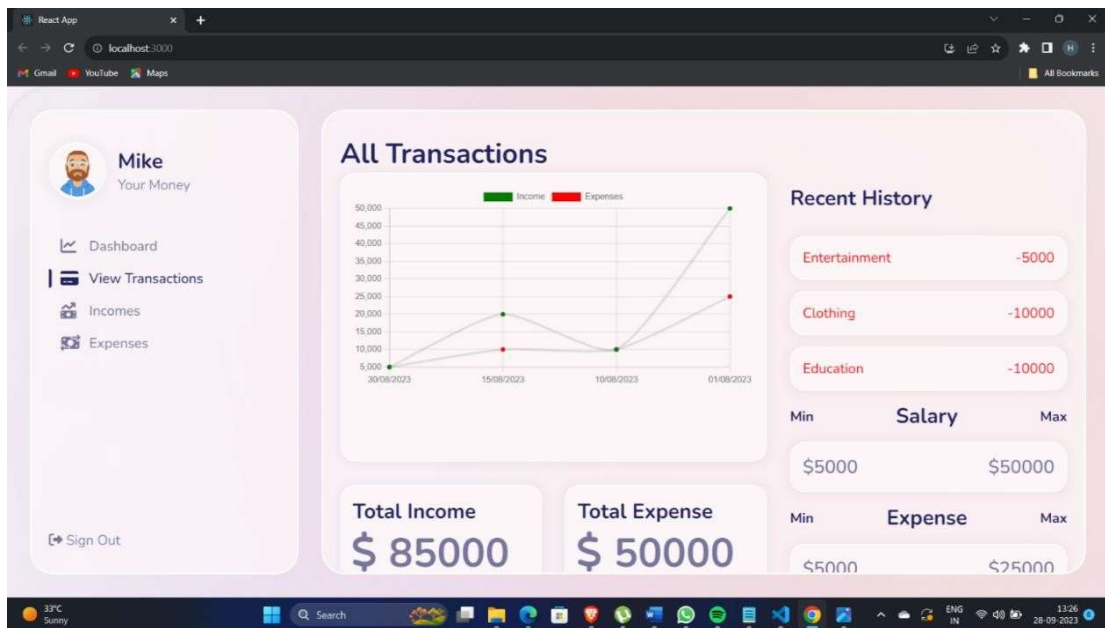


Fig.4.1.Graph

The screenshot shows the 'Add Expense' form in the financial dashboard. The form includes input fields for 'Expense Title', 'Expense Amount', and 'Enter A Date'. There is a 'Select Option' dropdown menu and an 'Add A Reference' text area. A red '+ Add Expense' button is at the bottom. On the right, there is a list of existing expenses with icons, dates, and descriptions. The total expense is shown as \$50,000.

Total Expense: \$50,000

Expense Title:

Expense Amount:

Enter A Date:

Select Option:

Add A Reference:

+ Add Expense

Existing Expenses:

- Entertainment: \$ 5000, 22/08/2023, Entertainment
- Clothing: \$ 10000, 13/08/2023, Clothing
- Education: \$ 10000, 06/08/2023, For Education
- Rent: \$ 25000, 02/08/2023, Home Rent

Fig.4.2.Output