

## Student Internship Manual 2024 – 25

GSFC University

# INDUSTRIAL INTERNSHIP REPORT

### BUDGET BUDDY

Submitted in partial fulfillment of the Requirements for the award of

### Degree of

##### Bachelor of Technology (B. Tech) in Computer Science and Engineering

Submitted By

##### **Nand Manish Patel**

##### **24bt04082**

**Computer Science and Engineering**

Submitted To

**GSFC University, Vadodara  
  
 Internship Institution:-   
 MAMO TECHNO LABS   
  
Internship Period: 2 JUNE - 2 JULY 2025  
  
Date of Report Submission:**

### **DECLARATION**

I hereby declare that the Industrial Internship Report entitled ("BUDGET BUDDY WEBSITE") is an authentic record of my own work as requirements of Industrial Internship during the period from **2 June**  to  **2 July** for the award of degree (B TECH CSE), GSFC University, Vadodara, under the guidance of **Ms. Neha Puajara**

**Nand Manish Patel  
 24BT04082**

**Date:**

##### **CERTIFICATE**



##### **ACKNOWLEDGEMENT**

I would like to express my heartfelt gratitude to all those who supported me throughout my internship and project journey.

First and foremost, I thank **GSFC University** and the **Department of Computer Science and Engineering** for providing me with the opportunity to pursue this internship project titled **“Expense Tracker using MERN Stack”**.

I sincerely thank my faculty coordinator **Ms. Neha Pujara**  for their valuable feedback, motivation, and constant support during the internship.

I am also grateful to my internship mentor(s) and team members for guiding me through the learning of the MERN stack and helping me understand real-world web development processes. Their insights and encouragement have helped me improve my technical and problem-solving skills.

Last but not least, I would like to thank my family and friends for their continuous support and encouragement throughout this internship.

##### TABLE OF CONTENT

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Pg. no.** |
| 1 | Declaration | 4 |
| 2 | Certificate | 5 |
| 3 | Acknowledgment | 6 |
| 4 | Table of Content | 7 |
| 5 | Abbreviations and Nomenclature (If any) |  |
| 6 | Chapter: 1 Introduction to Project | 8 |
|  | Chapter: 2 Major components of Project / Internship | 10 |
|  | Chapter: 3 Methodology adopted to carry out a project / Internship | 14 |
|  | Chapter: 4 Tools and Technology used | 17 |
|  | Chapter: 5 Data on the Project | 19 |
|  | Chapter: 6 Snapshots | 21 |
|  | Chapter: 7 Observations | 23 |
|  | Chapter: 8 Results and Discussions | 25 |
|  | Chapter: 9 Conclusion & Future scope | 27 |
| 7 | References | 29 |
| 8 | Data Sheet (if any) |  |
| 9 | Appendices (if any) |  |

**Chapter 1: Introduction to Project**

### **1.1 Background**

In today’s digital age, Expense Trackers play a vital role in communication, information sharing, and community building. Whether personal or professional, Expense Trackerging platforms serve as a medium for individuals and organizations to express ideas, publish articles, and engage with readers worldwide. Web technologies have advanced significantly, and full-stack development now allows developers to create dynamic, scalable, and interactive platforms efficiently.

This project focuses on building a Expense Tracker Website using the MERN stack—a combination of four powerful technologies: MongoDB, Express.js, React.js, and Node.js. The Expense Tracker platform allows users to create, read, edit, and delete Expense Tracker posts, register and log in securely, and interact with the application through a smooth and responsive interface.

### **1.2 Objective of the Project**

The main objective of this project is to design and develop a functional and responsive Expense Tracker application using full-stack web development technologies. Specific objectives include:

* To gain practical exposure to full-stack development using MERN.
* To implement user authentication and authorization using secure practices.
* To allow CRUD (Create, Read, Update, Delete) operations for Expense Tracker posts.
* To deploy the application on cloud platforms for live access.
* To document the learning process and development journey.

### **1.3 Problem Statement**

Many individuals and small businesses require simple, customizable Expense Tracker without relying on third-party CMS platforms like WordPress. Existing solutions may be complex or may not offer full control over the backend and frontend functionality. This project aims to bridge that gap by developing a lightweight, full-stack Expense Tracker website with essential features that allow:

* User registration and login functionality.
* Secure session handling and role-based access control.
* A clean interface to write, edit, and publish posts.
* Dynamic display of Expense Tracker posts from a database.

### **1.4 Scope of the Project**

This project encompasses both frontend and backend development:

* Frontend (React.js): Creating a responsive user interface with page routing, forms, and post display.
* Backend (Node.js & Express.js): Building RESTful APIs to handle data operations and user authentication.
* Database (MongoDB): Storing Expense Tracker posts and user information in a scalable, NoSQL cloud database.
* Deployment: Hosting the whole website on Vercel for global accessibility.

While this project does not currently include features like likes, comments, or admin dashboards, it is designed in a scalable way so that such features can be added in the future.

### **1.5 Methodology** The development process followed a weekly milestone-based structure, combining theoretical learning with hands-on implementation. The methodology included:

* Setting up the development environment with necessary tools.
* Learning individual MERN stack technologies through practical use.
* Designing and structuring frontend components and backend APIs.
* Testing features locally and integrating frontend with backend.

.

**Chapter 2: Major Components of Project / Internship**

### **2.1 Introduction**

The development of a full-stack Expense Tracker website involves the integration of multiple technologies across both frontend and backend layers. The MERN stack — consisting of MongoDB, Express.js, React.js, and Node.js — is a widely adopted technology stack that enables end-to-end development using JavaScript. Each component plays a crucial role in building and delivering a scalable and responsive web application.

This chapter provides a detailed explanation of each major component used in the project and how they collectively contribute to the overall architecture of the application.

### **2.2 React.js – The Frontend Library**

React.js is a powerful JavaScript library developed by Facebook, used to build interactive and dynamic user interfaces for web applications.

#### Key Features:

* Component-Based Architecture: UI is broken into reusable components (e.g., Navbar, StatsCard, Footer).
* Virtual DOM: Ensures faster rendering by updating only the parts of the UI that change.
* One-Way Data Binding: Promotes predictable UI state management.
* React Router: Used for managing page navigation without refreshing the browser.

In this project, React handles the Expense Tracker homepage, user registration/login forms, Expense Tracker creation, and post display interfaces.

### **2.3 Node.js – The Server Environment**

Node.js is a JavaScript runtime built on Chrome's V8 engine that allows developers to run JavaScript on the server side.

#### Key Features:

* Asynchronous and Event-Driven: Suitable for real-time applications.
* Lightweight and Fast: Ideal for microservices and REST APIs.
* NPM (Node Package Manager): Offers access to thousands of open-source libraries.

Node.js powers the backend server for this Expense Tracker project, handling requests, processing data, and sending responses.

### **2.4 Express.js – The Web Framework**

Express.js is a minimal and flexible web framework for Node.js that simplifies the creation of APIs and server-side logic.

#### Key Features:

* Routing: Easily maps HTTP requests to specific handlers.
* Middleware Support: Allows adding custom logic at various stages of request handling.
* RESTful API Creation: Simplifies CRUD operations on Expense Tracker and user data.

In the project, Express is used to define API routes such as /register, /login, /create, /post/:id, /edit, and /delete.

### **2.5 MongoDB – The Database**

MongoDB is a NoSQL, document-oriented database that stores data in JSON-like formats (BSON).

#### Key Features:

* Flexible Schema: Allows dynamic document structures.
* High Performance & Scalability: Ideal for web applications with growing data.
* Cloud-Hosted (MongoDB Atlas): Enables easy deployment and remote access.

All Expense Tracker posts and user data are stored in collections in MongoDB. It integrates with the backend via Mongoose (ODM).

### **2.6 Mongoose – Object Data Modeling (ODM)**

Mongoose is a Node.js library that provides schema-based solutions for modeling and validating MongoDB data.

#### Uses in the Project:

* Define Expense Tracker and User schemas
* Perform validations (e.g., required fields)
* Simplify database operations (e.g., .find(), .save(), .updateOne())

It ensures that the data stored in the database is well-structured and consistent.

### **2.7 Axios – HTTP Client**

Axios is a promise-based HTTP client used to make API calls from the frontend to the backend.

#### Uses:

* Sending POST requests for login and registration
* Fetching Expense Tracker data to display on the homepage
* Sending PUT and DELETE requests for editing or deleting Expense Trackers

Axios provides error-handling capabilities and is essential for frontend-backend communication.

### **2.8 React Context API – State Management**

React Context API enables passing data through the component tree without prop drilling.

#### Usage in the Project:

* Stores login state and user info
* Controls access to protected pages
* Dynamically shows/hides navigation items like “Create Post” or “Logout”

This improves user experience by managing session-related UI states globally.

### **2.9 Deployment Tools: Vercel**

* Vercel: Used for hosting the frontend React application.

These platforms provide continuous deployment, environment variable management, and public access to the project from any device.

### **2.10 Git and GitHub – Version Control**

Git is a version control system, and GitHub is a cloud-based platform for hosting and collaborating on code.

#### Uses:

* Tracking project progress
* Pushing regular commits
* Sharing source code via a GitHub repository

This ensured the project remained organized and version-controlled throughout the internship period.

### **2.11 Summary**

Each component of the MERN stack and supporting tools played a vital role in the successful development of the Expense Tracker website. From frontend interactivity to backend logic and data persistence, the integration of these technologies offered a real-world full-stack development experience

# 

# Chapter 3: Methodology Adopted to Carry Out the Project / Internship

### **3.1 Introduction**

Methodology refers to the systematic process followed to successfully complete a project. A well-structured methodology ensures clarity in execution, time-bound progress, and measurable learning outcomes. During the internship, a combination of **learning by doing**, **milestone-based planning**, and **agile-inspired iterative development** was followed to build a fully functional Expense Tracker website using the MERN stack.

This chapter outlines the strategy adopted, weekly planning, learning cycle, tools used, and execution approach that ensured timely and structured progress throughout the internship.

### **3.2 Development Approach**

The project was executed using a **step-by-step incremental development model**, aligned with weekly milestones. The development lifecycle consisted of the following stages:

1. **Requirement Understanding and Planning**
2. **Learning Core Concepts and Tools**
3. **Frontend Development**
4. **Backend Development**
5. **Integration of Frontend and Backend**
6. **Authentication and Authorization**
7. **Testing and Debugging**
8. **Deployment and Documentation**

### 

### **3.3 Learning and Implementation Cycle**

The learning and implementation process followed a **daily cycle**, where each day included:

* **Concept Learning:** Watching tutorials or reading documentation to understand specific technologies or tools.
* **Hands-On Practice:** Writing code and testing snippets locally.
* **Feature Implementation:** Applying learned concepts to build specific parts of the project.
* **Problem Solving:** Debugging errors, searching solutions, and seeking mentorship if needed.

This cycle allowed a balance between gaining theoretical knowledge and practical skills.

### **3.4 Weekly Milestone Structure**

The entire internship duration was divided into **4 weeks**, each with clear goals:

* **Week 1:** Setting up the development environment, learning MERN basics, and starting frontend pages like homepage, login, and registration.
* **Week 2:** Backend setup with Node.js and Express.js, connecting to MongoDB Atlas, and integrating API calls with the frontend.
* **Week 3:** Building authentication system using JWT, handling protected routes, and implementing full Expense Tracker post CRUD features.
* **Week 4:** Finalizing features, UI polish, deploying backend and frontend using Render and Vercel, and preparing project documentation.

### 

### **3.5 Tools and Platforms Used**

|  |  |
| --- | --- |
| **Tool/Platform** | **Purpose** |
| **VS Code** | Writing and editing code |
| **Git & GitHub** | Version control and project backup |
| **MongoDB Atlas** | Cloud database for data storage |
| **Render** | Backend deployment |
| **Vercel** | Frontend deployment |
| **Postman** | API testing |
| **Axios** | API calls from React frontend |
| **React Router** | Routing between different pages |
| **Mongoose** | Schema creation and database management |

### **3.6 Testing and Debugging Strategy**

Testing was done using:

**Postman:** For backend API testing

**Browser DevTools:** To inspect and debug frontend behavior

**Console Logging**: For tracking data flow and identifying issues

**Manual Functional Testing:** To validate user flow from login to creating/editing posts

Errors like CORS issues, form validation failures, and token expiration were resolved during this phase.

### 

### **3.7 Deployment and Final Review**

The final deployment involved:

* Hosting the backend API on **Render**, connecting it with MongoDB Atlas using environment variables.
* Deploying the frontend on **Vercel**, linking it to the hosted backend API.
* Performing final checks for mobile responsiveness, API integration, and security validation.

Once deployed, screenshots were taken, and the GitHub repository link and live site URL were documented for submission.

**Chapter 4: Tools and Technology Used**

### **4.1 Introduction**

Building a Expense Tracker website using the MERN stack requires a combination of programming tools, development environments, libraries, databases, and hosting platforms. Each tool or technology used during the internship played a specific role in ensuring the successful design, development, testing, and deployment of the project. This chapter provides a comprehensive overview of the tools and technologies utilized throughout the internship period.

### **4.2 Technologies Used**

|  |  |
| --- | --- |
| **Technology** | **Description** |
| **MongoDB** | A NoSQL document-oriented database used to store Expense Tracker posts and user data in JSON-like format. Hosted on MongoDB Atlas. |
| **Express.js** | A minimal and flexible Node.js web application framework used for building RESTful APIs. |
| **React.js** | A JavaScript library for building user interfaces. It helps in creating reusable UI components and managing state. |
| **Node.js** | A server-side JavaScript runtime that allows backend development using JavaScript. It powers the Express server. |
| **Mongoose** | An Object Data Modeling (ODM) library for MongoDB and Node.js, used to define schemas and interact with the database. |
| **Axios** | A promise-based HTTP client used in React to make asynchronous API requests to the backend server. |
| **React Router DOM** | A routing library for React that manages navigation between different components/pages. |

### 

### **4.3 Development Tools**

|  |  |
| --- | --- |
| **Tool** | **Purpose** |
| **Visual Studio Code (VS Code)** | Source-code editor used to write and manage both frontend and backend code efficiently. |
| **Postman** | A testing tool used for API development. It allows testing of all backend routes like login, register, and CRUD operations. |
| **Git** | A version control system used to track code changes, manage branches, and collaborate with repositories. |
| **GitHub** | A cloud-based platform to host and share code, manage commits, and store project files securely. |
| **npm (Node Package Manager)** | Used to install libraries and dependencies such as Express, Mongoose, and React packages. |

### **4.4 Deployment Platforms**

|  |  |
| --- | --- |
| **Platform** | **Use** |
| **Render** | A cloud platform used to host the Node.js backend server and expose the API endpoints to the internet. |
| **Vercel** | A frontend deployment platform used to host the React application and connect it to the backend API. |
| **MongoDB Atlas** | A cloud-hosted version of MongoDB used for creating and managing the database online. |

### **4.5 Other Utilities**

|  |  |
| --- | --- |
| **Utility** | **Description** |
| **Environment Variables (.env)** | Used to securely store sensitive configuration values like database URLs, API keys, and JWT secrets. |
| **Browser Developer Tools** | Built-in browser tools (in Chrome/Firefox) used to debug frontend issues, inspect elements, and monitor network requests. |

# Chapter 5: Data on the Project

### **5.1 Introduction**

Data is a crucial element in any web application. In this Expense Tracker website project, data plays a central role in representing users, Expense Tracker posts, and user interactions. This chapter discusses the types of data used, their structure, and how they are processed within the system using the MERN stack.

### **5.2 Types of Data**

The primary categories of data managed in this project are:

1. **User Data**
2. **Expense Tracker Post Data**
3. **Session and Authentication Data**

These data types are created, read, updated, or deleted through various frontend and backend operations using forms, API calls, and database queries.

### **5.3 Data Flow in the Application**

The flow of data in the Expense Tracker website is as follows:

1. **User submits form data** (e.g., login, register, Expense Tracker post) on the frontend.
2. Data is sent to the backend API using **Axios** via HTTP requests (POST/GET/PUT/DELETE).
3. **Express.js** handles the request and validates the data.
4. Valid data is processed and stored or retrieved from **MongoDB** using **Mongoose**.
5. The backend returns a response, which is used by **React** to update the UI dynamically.

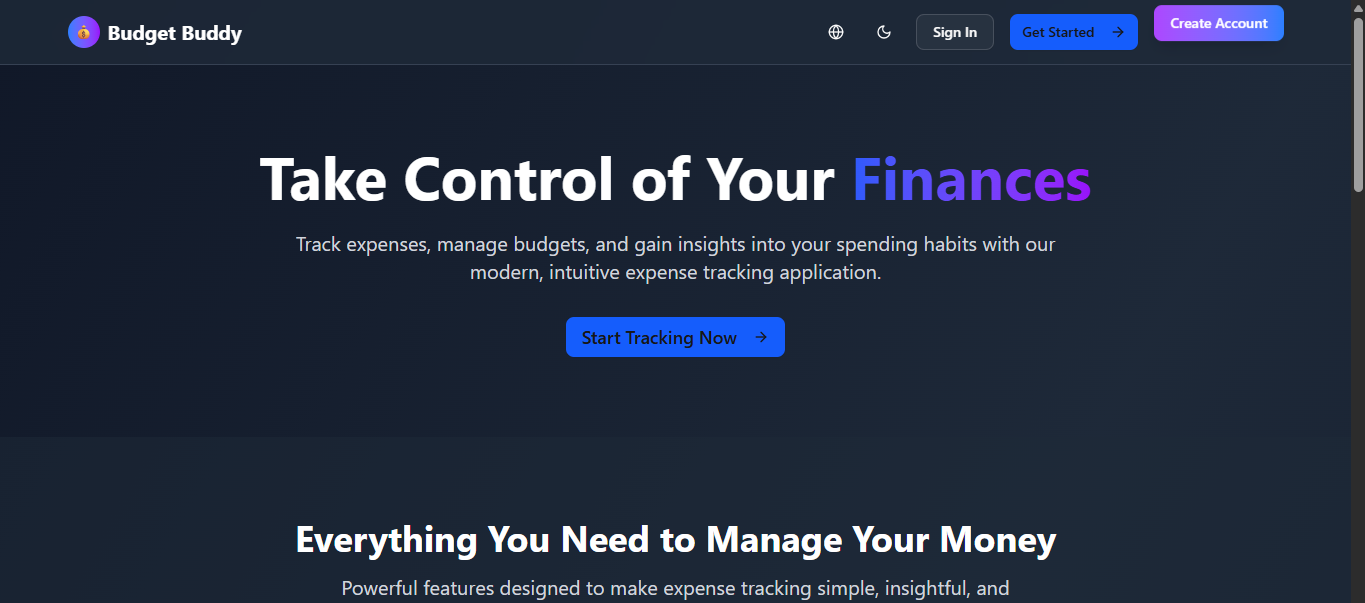
### **5.6 Data Validation and Security**

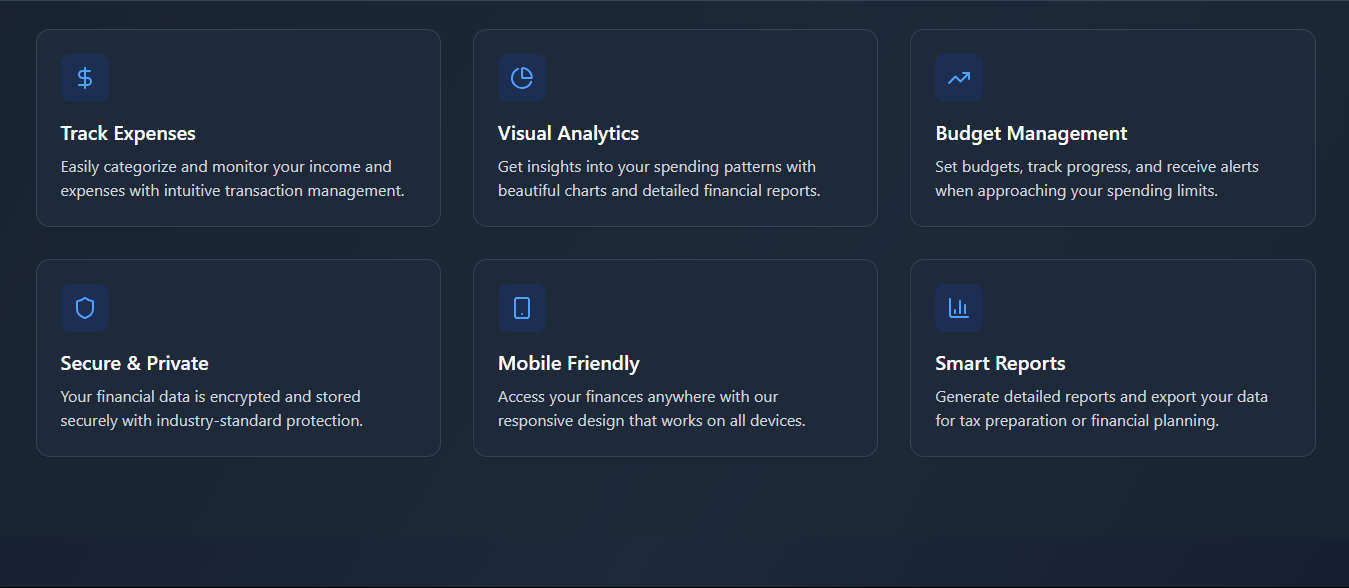
* **Form Validation:** Handled in React using required fields and alerts.
* **Schema Validation:** Mongoose ensures only correctly formatted data is saved.
* **Password Encryption:** Passwords are encrypted using **crypt** before being stored.
* **Token-based Security:** JWT tokens ensure only authenticated users can create/edit expenses.

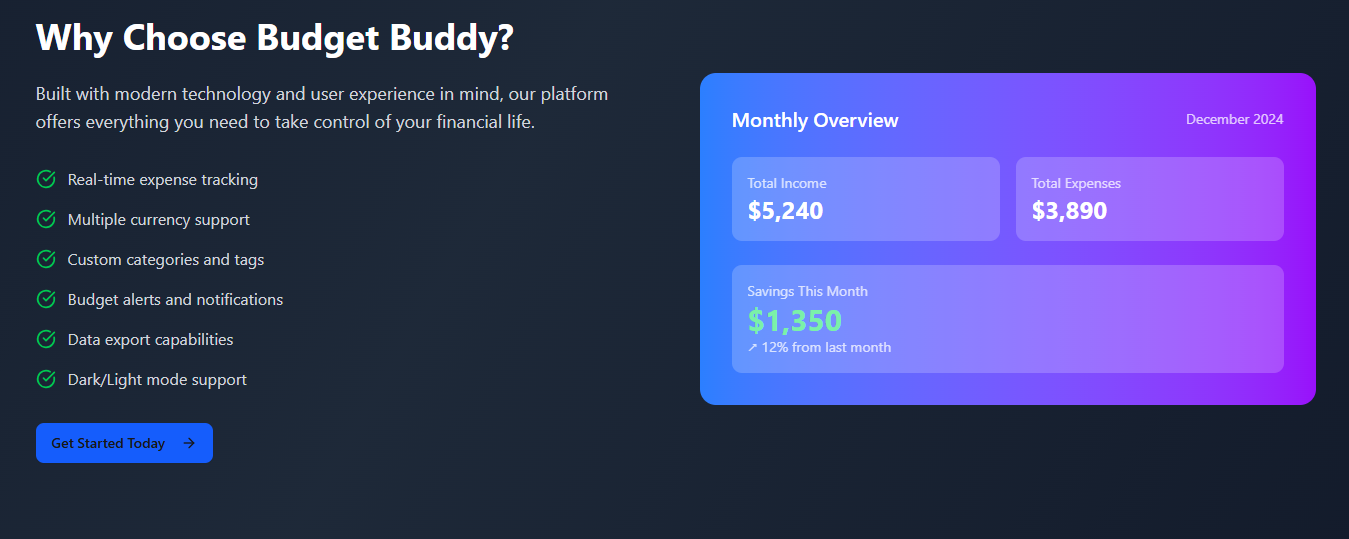
# 

# Chapter 6: Snapshots

**Home Page/ Dashboard**







# 

# 

# 

# 

# Chapter 7: Observation

### **7.1 Introduction**

The internship provided a valuable opportunity to engage with real-world web development using the MERN stack. Throughout the four-week period, consistent learning, implementation, and problem-solving helped in gaining both technical skills and project execution experience. This chapter records the key observations made during different stages of the project.

### **7.2 Technical Observations**

1. **Integration Across Technologies Is Crucial** Although React, Node.js, and MongoDB are powerful individually, integrating them smoothly using Express.js and API communication was key to building a functional full-stack application.
2. **Asynchronous Programming is Fundamental** Using asynchronous functions like async/await and handling promises in Axios was critical when working with real-time data fetching and user actions.
3. **State Management Simplifies UI Control** React's useState and useContext hooks made it easier to manage user sessions and dynamically update the UI based on login status or form data.
4. **Security and Validation Are Must-Haves** Observed that without proper data validation, both frontend and backend become vulnerable. Implementing validation and JWT-based authentication greatly improved the project's security layer.
5. **Deployment Comes With Its Own Challenges** Hosting the application revealed deployment-specific issues like CORS errors, environment variable handling, and database connectivity — which had to be debugged carefully.

### **7.3 Learning Process Observations**

1. **Consistent Daily Practice Boosted Confidence** Following a structured daily plan (learning and building) helped stay consistent and motivated during the internship.
2. **Understanding Before Copying is Essential** It was noticed that blindly copying code didn’t help in the long run. Taking time to understand the logic made debugging and modification easier later.
3. **Debugging is a Learning Opportunity** Many unexpected errors during API integration or form submission helped in learning tools like browser dev tools, Postman, and network inspectors.
4. **Documentation Helps Future Maintenance** Writing a clean README, comments, and keeping logs made the codebase more maintainable and easier to revise even after weeks.

### **7.4 Soft Skills & Behavioral Observations**

1. **Time Management Was Crucial** Dividing the entire internship period into week-wise and day-wise tasks helped in completing the project on time without feeling overwhelmed.
2. **Self-Learning is Empowering** Observed that relying on self-learning through video tutorials, documentation, and experimentation builds confidence and independence.
3. **Communication & Reporting Are Important** Regularly updating logs and preparing reports improved written communication skills — a crucial aspect for any developer or team member.

### **7.5 Summary**

The internship allowed observation and experience beyond just coding — including planning, researching, debugging, testing, deploying, and documenting a full project cycle. These observations are not only technically insightful but also serve as foundational knowledge for future internships, placements, and real-world project development. The MERN stack has proven to be a powerful and practical technology combination for modern web applications.

# Chapter 8: Results and Discussions

### **8.1 Introduction**

This chapter presents the results of the four-week internship project focused on developing a Expense Tracker website using the MERN stack. It evaluates how well the objectives were achieved, discusses the project’s performance, highlights working features, and reflects on potential improvements and future enhancements.

### **8.2 Results of the Internship Project**

The key deliverables and outcomes of the project are summarized below:

|  |  |
| --- | --- |
| **Objective** | **Result Achieved** |
| **Learning MERN Stack** | Successfully learned React.js, Node.js, Express.js, and MongoDB, along with supporting tools like Mongoose and Axios. |
| **User Authentication** | Fully functional login, registration, session handling using JWT (JSON Web Tokens), and protected routes implemented. |
| **Expense Tracker Post CRUD Features** | Users can create, read, update, and delete Expense Tracker posts. Data is stored and retrieved from MongoDB using Mongoose models. |
| **Frontend Development** | Built using React with dynamic rendering, form handling, routing, and responsive design. |
| **Backend API Integration** | Created RESTful APIs using Express.js, connected to the frontend via Axios, and tested via Postman. |
| **Deployment** | Successfully deployed backend on **Render**, frontend on **Vercel**, and database on **MongoDB Atlas**. |
| **Code Documentation** | Proper code structure, comments, README file, and version control using GitHub were maintained. |

### **8.3 Performance of the Application**

The final deployed Expense Tracker website includes the following working features:

* **User Registration & Login:** Authenticates new and existing users, using encrypted passwords.
* **JWT-based Authentication:** Maintains secure user sessions and protects routes like “Create Post” and “Edit Post”.
* **Expense Tracker Post Creation & Management:** Users can add, edit, and delete posts they authored.
* **Responsive UI:** Optimized for both desktop and mobile devices using CSS.
* **Live Deployment:** Available online for access via any browser using hosted URLs.

### **8.4 Challenges Faced During Implementation**

|  |  |
| --- | --- |
| **Challenge** | **Discussion** |
| **CORS Errors During API Calls** | Occurred when integrating frontend and backend on different domains. Solved using CORS middleware in the Express server. |
| **Authentication Persistence** | Maintaining user login state across refresh required use of React Context and session storage. |
| **Routing Conflicts** | Conflicts between client-side and server-side routing were resolved by adjusting deployment and using HashRouter. |
| **Form Validation** | Implemented both frontend (React) and backend (Express) validation to ensure data quality. |
| **Deployment Delays** | Environmental variable misconfigurations initially caused deployment failures, which were resolved by reviewing platform documentation. |

### **8.5 Discussions & Insights**

1. **Real-World Simulation:** The project simulated a real-world web development environment involving frontend-backend communication, database integration, and secure user authentication.
2. **Hands-on Learning:** Working with real tools like GitHub, Postman, Render, and Vercel enhanced technical readiness for future internships and jobs.
3. **Full Cycle Experience:** From project setup to deployment, the internship provided an end-to-end development experience, including debugging, documentation, and reporting.
4. **Scope for Future Enhancements:**
   * Adding features like comments, categories, and search functionality.
   * Creating an admin panel for Expense Tracker moderation.
   * Enhancing UI/UX with modern frameworks like Tailwind CSS or Material UI.

# Chapter 9: Conclusion & Future Scope

**9.1 Conclusion**

The internship journey of developing a **Expense Tracker Website using the MERN Stack** proved to be a valuable and transformative learning experience. Over the course of four weeks, a strong foundation was established in full-stack web development using **MongoDB, Express.js, React.js, and Node.js**.

The internship successfully blended theoretical learning with practical application. The Expense Tracker platform developed as part of this project enables users to register, log in securely, create posts, view Expense Trackers, edit and delete their entries — fulfilling all core CRUD functionalities. Additionally, the application was deployed using **Vercel**, ensuring real-time access and remote usability.

Key takeaways from the project include:

* Understanding how the frontend and backend communicate through RESTful APIs.
* Implementing secure authentication using JWT tokens.
* Managing state effectively in React using Context API and hooks.
* Hosting and deploying real-world applications using cloud services.
* Practicing code version control using Git and GitHub.

The project not only enhanced technical competence but also improved critical thinking, debugging, time management, and problem-solving abilities. It provided a practical glimpse into how modern web applications are built and maintained in industry settings.

### **9.2 Future Scope**

While the current version of the Expense Tracker application meets essential requirements, there are several areas for enhancement and future development:

#### 🔹 1. Comments & Likes Feature

Allow users to comment on Expense Tracker posts or react with likes/dislikes to increase interactivity.

#### 🔹 2. Rich Text Editor

Replace the basic textarea with a rich text editor (like Quill.js) to allow users to format their Expense Tracker content with headings, lists, links, etc.

#### 🔹 3. Admin Dashboard

Introduce an admin panel to manage users, moderate content, and generate analytics about Expense Tracker activity.

#### 🔹 4. Categories & Tags

Add filtering capabilities by categorizing Expense Trackers with topics or keywords to enhance searchability.

#### 🔹 5. Pagination & Search

Implement pagination for Expense Tracker lists and add search functionality to allow users to quickly find specific posts.

#### 🔹 6. Image Upload Support

Enable users to upload and embed images in Expense Tracker posts using cloud storage (e.g., Cloudinary or Firebase).

#### 🔹 7. Mobile Optimization

Further improve responsiveness and performance for users accessing the site on various mobile devices.

#### 🔹 8. Notification System

Add notifications for user actions such as successful login, post creation, or when someone comments on a post.

### 

### **9.3 Final Thoughts**

This internship has provided a strong stepping stone into the world of full-stack web development. The skills and knowledge acquired through this hands-on experience will be extremely useful in future academic projects, industry internships, and placements. The project stands as a complete, functional portfolio piece and can be expanded further as time and needs evolve.

The experience gained here proves that with structured planning, consistent effort, and curiosity to explore, even complex technologies like the MERN stack can be mastered and implemented effectively.

# References

Below is the list of resources, tools, and platforms referred to during the course of the internship project for learning, development, debugging, and deployment:

### **Official Documentation**

* React.js Documentation – https://reactjs.org/docs
* Node.js Documentation – https://nodejs.org/en/docs
* Express.js Guide –<https://expressjs.com>
* MongoDB Documentation –<https://www.mongodb.com/docs>
* Mongoose Guide – https://mongoosejs.com/docs
* React Router –<https://reactrouter.com>
* Axios –<https://axios-http.com>

### 

### **Tools and Platforms**

* Visual Studio Code –<https://code.visualstudio.com>
* Postman API Testing –<https://www.postman.com>
* Git & GitHub –<https://github.com>
* Render Deployment –<https://render.com>
* Vercel Hosting –<https://vercel.com>
* MongoDB Atlas –<https://www.mongodb.com/cloud/atlas>

### **Other Learning Resources**

* Stack Overflow –<https://stackoverflow.com>
* MDN Web Docs –<https://developer.mozilla.org>
* W3Schools (for general HTML/CSS/JS help) –<https://www.w3schools.com>