**PROJECT SYNOPSIS REPORT**

**ON**

**EXPENSE TRACKER**

**SUBMITTED**

**TO**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**FOR**

**Backend Engineering (23CS008)**



Submitted To:- Submitted By:-

Mr. Rahul. Tanisha Bhardwaj (2310991065).

Tushar Koundal (2310991071).

Upasvi (2310991072).

# Index

Sr. no Topic Page No

1. Problem Statement 3
2. Title of project 3
3. Objective & Key Learning’s 3
4. Options available to Execute the project 3
5. Tech Stack 4
6. Advantages/ Disadvantages 4
7. Implementation Strategy 5
8. Conclusion 5
9. References 5

### **Problem Statement**

In today’s fast-paced world, individuals and small businesses often struggle to manage daily expenses and maintain financial discipline. Manual methods like notebooks or spreadsheets are prone to errors, time-consuming, and lack insights into spending patterns. Existing apps may be too complex, expensive, or fail to provide real-time insights.

The **Expense Tracker** project addresses these challenges by providing a **secure, user-friendly, and scalable platform** to track expenses, categorize spending, generate reports, and visualize trends. It enables users to maintain financial awareness, plan budgets efficiently, and make data-driven decisions to improve savings and reduce unnecessary spending.

### **Title of Project**

**Expense Tracker**

### **Objective & Key Learnings**

* Develop a user-friendly system for tracking daily expenses and income.
* Implement categorization, reporting, and dashboard analytics for better financial insights.
* Integrate secure user authentication and data storage.
* Learn and apply full-stack development using **Next.js, Node.js, and MongoDB**.
* Understand cloud deployment, CI/CD, and performance optimization.
* Enhance skills in responsive UI/UX and data visualization using charts.

### **Options Available to Execute the Project**

* **Serverless Architecture:** Cost-effective, scalable, and minimal server management.
* **Monolithic or Modular Backend:** Simple monolithic backend for a small user base or modular design for scalability.
* **Frontend Frameworks (Next.js, React.js):** Accelerates development and ensures responsive UI.
* **Backend-as-a-Service (BaaS):** Firebase or MongoDB Atlas for rapid backend deployment.
* **Data Visualization Libraries (Chart.js, D3.js):** For real-time expense analysis and dashboards.
* **CI/CD Tools (GitHub Actions):** Automates testing and deployment for continuous updates.

### **Tech Stack**

**Frontend** – Next.js, React.js, Tailwind CSS

* Performance: Server-side rendering ensures fast page load.
* Scalability: Supports growing user base.
* Flexibility: Easily integrates with backend APIs and databases.

**Backend** – Node.js, Express.js

* Efficient handling of API requests and CRUD operations.
* Secure authentication using JWT/Firebase.

**Database** – MongoDB

* Flexible NoSQL storage for user and expense data.
* Schema management using Mongoose.

**Optional Analytics/ML Layer**

* Chart.js / D3.js for dashboards.
* TensorFlow/PyTorch for predictive expense insights (advanced feature).

**Deployment & Hosting** – Vercel (frontend), AWS/DigitalOcean (backend), MongoDB Atlas (database)

### **Advantages / Disadvantages**

**Advantages:**

1. **Financial Awareness:** Helps users track expenses and manage budgets efficiently.
2. **Automation:** Automatically categorizes expenses and generates real-time reports.
3. **Accessibility:** Cloud-based, accessible on multiple devices anytime.
4. **Scalable:** Supports multiple users and large datasets.
5. **Data Security:** Implements encryption and secure authentication.

**Disadvantages:**

1. **Initial Setup:** Requires initial configuration for user accounts and cloud deployment.
2. **Data Dependency:** Accurate insights rely on consistent and correct user input.
3. **Limited AI Features (Optional):** Advanced predictive analytics require additional resources.

### **Implementation Strategy**

1. **Backend Development**
   * Set up **Node.js and Express.js** server for API handling.
   * Design endpoints for CRUD operations on expenses and categories.
   * Use MongoDB for secure storage of user data and expense records.
2. **Frontend Development**
   * Build responsive UI with **Next.js and Tailwind CSS**.
   * Integrate API calls to fetch, update, and display expense data.
   * Implement dynamic dashboards for charts and analytics.
3. **Authentication & Security**
   * Use **JWT/Firebase** for secure user login and session management.
   * Implement SSL/TLS for encrypted communication.
   * Validate and sanitize inputs to prevent attacks like XSS or injection.
4. **Analytics & Reporting**
   * Use **Chart.js/D3.js** to visualize expenses by category, month, and payment type.
   * Generate downloadable CSV or PDF reports for offline use.
5. **Testing & Deployment**
   * Perform unit, integration, and performance testing.
   * Deploy frontend on **Vercel**, backend on **AWS/DigitalOcean**, and database on **MongoDB Atlas**.
   * Monitor app performance and optimize API response times.

### **Conclusion**

The Expense Tracker project provides a **user-friendly, secure, and scalable platform** for managing personal and small business finances. By automating expense tracking, categorization, and reporting, it empowers users to maintain financial discipline, make informed decisions, and achieve their budgeting goals. The integration of modern full-stack technologies ensures a responsive, reliable, and future-proof system.

### **References**

1. <https://nodejs.org/en>
2. <https://react.dev/>
3. <https://expressjs.com/>
4. https://tailwindcss.com/docs