Technical Architecture Document

Project: Cost Wise Innovators – Cost Optimization Dashboard
Team: CSDD1
Date: April 1, 2025

---

1. Introduction

This document provides a high-level and low-level overview of the technical architecture for the Cost Wise Innovators Dashboard. It outlines the components, technology stack, system workflow, and data flow to support development and deployment.

---

2. System Overview

The Cost Optimization Dashboard is a full-stack web application built with React.js (frontend), Node.js and Express.js (backend), and Firebase/MongoDB (database). The system supports user authentication, data visualization, and cost analytics.

---

3. Architecture Design

The system follows a modular, component-based architecture with separation of concerns:

- Frontend (Client Side): React.js SPA
- Backend (Server Side): Node.js + Express.js RESTful APIs
- Database: Firebase Realtime DB / MongoDB Atlas (cloud-hosted)
- Authentication: Firebase Auth / Auth0
- Hosting & Deployment: Vercel / Firebase Hosting / Netlify

---

4. Component Diagram Description

- **Frontend UI**: Handles user interaction, routing, and visual rendering
- **API Layer**: Intermediary between frontend and database; handles business logic
- **Database**: Stores user profiles, expense records, and budget settings
- **Authentication Service**: Manages user sessions, password reset, role management

---

5. Data Flow Diagram (Level 1)

1. User logs in → Auth service verifies credentials
2. Upon login → Dashboard data fetched via API
3. User performs actions (add expense, set budget) → Request sent to backend
4. Backend updates database → Returns updated data to frontend
5. UI re-renders charts and budget progress in real-time

---

6. Technology Stack

- **Frontend**: React.js, JavaScript, HTML5, CSS3
- **Backend**: Node.js, Express.js
- **Database**: MongoDB Atlas or Firebase Realtime Database
- **Authentication**: Firebase Authentication or Auth0
- **Deployment**: Vercel / Netlify / Firebase Hosting
- **Version Control**: Git & GitHub
- **Visualization Library**: Chart.js or D3.js
- **Design Tools**: Figma / Adobe XD

---

7. Security Considerations

- All data transactions use HTTPS
- Authentication uses JWT / OAuth2 mechanisms
- Input validation and sanitation on both frontend and backend
- Role-based access control (optional future enhancement)
- Regular backups for database

---

8. Scalability & Performance

- Asynchronous API calls with loading indicators
- Efficient state management on frontend (e.g., Redux or Context API)
- Backend supports concurrent requests using Node.js non-blocking I/O
- Cloud-hosted database for horizontal scalability

---

9. Limitations

- No live bank integration in current version
- AI recommendations are not yet implemented
- Limited admin features (planned in future versions)

---

10. Conclusion

The proposed architecture is lightweight, scalable, and flexible for future upgrades. It aligns with modern full-stack web development practices and ensures a secure, responsive, and engaging experience for users.
