Expense Splitter Service – Project Report

Managing shared expenses among friends or roommates often leads to confusion, errors, and awkward money conversations. **Expense Splitter Service** solves this problem by providing an intelligent, automated way to track and settle shared costs fairly. It's a full-stack web application that allows users to create groups, add shared expenses, and automatically calculate individual balances. The system generates a minimal settlement plan showing the fewest payments required to balance all debts — ensuring quick and fair settlements for everyone. Built with a modern MERN stack (MongoDB, Express.js, React.js, Node.js) and Clerk authentication, the app ensures a smooth, secure, and responsive experience.

Tech Stack Used

- Frontend: React + Vite, Tailwind CSS, Clerk for authentication, Vercel for hosting
- Backend: Node.js + Express, Clerk middleware, deployed on Render
- Database: MongoDB (Atlas) with Mongoose ORM
- AI/Dev Tool: Gemini CLI for AI-assisted development
- Version Control: Git + GitHub

Development Journey

1. Read and understood the problem statement using ChatGPT explanations.
2. Asked ChatGPT for a MERN stack roadmap and started backend development.
3. Encountered prompt limits and discovered Gemini CLI via Buzz2Day YouTube channel.
4. Used Gemini CLI — uploaded full prompt describing strategy; it generated working prototype.
5. Configured environment variables (.env) with Clerk, MongoDB, and Render/Vercel keys.
6. Built core features: create/join groups, add/rename/delete groups, add expenses, compute balances.
7. Debugged a persistent backend error for 2 hours and resolved it.
8. Improved UI/UX with Tailwind styling and responsiveness.
9. Added logic to delete a user only if no amount was pending.
10. Deployed backend on Render and frontend on Vercel, with repository on GitHub.

About Gemini CLI:
Gemini CLI is Google's AI-powered command-line tool that brings Gemini model capabilities directly into the developer terminal. It can read and reason about codebases, generate files, fix bugs, run scripts, and interact with local tools using an intelligent "reason-and-act" loop. Developers can prompt it to perform end-to-end tasks such as scaffolding projects, debugging, and testing. It integrates tightly with local or remote environments via the Model Context Protocol (MCP), making AI-assisted coding seamless and fast.

Enhancements for Future Improvement

- Add unit and integration tests (Jest, Supertest).
- Implement role-based permissions (admin/member).
- Add transaction receipts and expense history export.
- Enable multi-currency support and currency conversion.
- Optimize settlement algorithm using graph-based approach.
- Introduce notifications/reminders for pending settlements.

- Integrate Sentry for monitoring and better error handling.
- Add pagination, rate limiting, and performance indexing in MongoDB.
- Improve UI/UX responsiveness and accessibility.

Challenges Faced

- Limited prompt length when using AI tools, solved with Gemini CLI.
- Environment variable setup for authentication and database access.
- Debugging a persistent backend logic error for hours.
- Ensuring authorization for protected routes using Clerk.
- Handling edge cases in expense splitting and rounding errors.
- Deployment configuration for different platforms (Render, Vercel).