I'll be starting my expense splitter for shared living project and i'll be using html,css,bootstrap,ejs,node.js,express.js and mongoDB.

Below are the features i'm going to build :
Features:
1. Budget Tracking: Submitting the total income per month. Submitting all the dues and to be receive payments.
2. Payment Sharing:
The Admin will add members to the split (Name and Contact Number). The members will list the overall amount of a particular thing. In case of an unequal split, there will be an option in the last to list the amount to a particular member. After the split and re-viewing by the admin, there will be an option to download the whole split in a PDF document.
3. Displaying of due bills: Particular section in the left-sided menu bar. With the amount and the name of the bill.
4. Dashboard of Overall evaluation: Amount to be received, Payments to make, Bills to be paid, and Number of bills to be paid. Total influx and deflux this month.

tell me the workflow step by step in easiest way. Like which page i have to build first and then after and all setups dont give me the code just workflow

Start with a minimal skeleton (auth, layouts, navigation), then layer features in small, testable slices. Build the core data models early (Users, Groups/Members, Expenses, Bills, Budgets, Settlements), wire basic CRUD, and only then add dashboards, due-bills sidebar, and PDF export. This order reduces rework and clarifies data flows for unequal splits and monthly tracking.[1] [2] [3]

Project setup

- Initialize Node.js, Express, EJS, MongoDB connection, and Bootstrap static assets; set up environment variables and folder structure (routes, controllers, models, views, public) for a clean baseline before features.[4] [1]
- Add auth scaffolding (signup, login, logout) with JWT/cookies and role support (admin vs member) so protected routes and admin review are possible from day one.[5] [1]

Data modeling first

- Define schemas: User, Group, Member (name, phone), Expense (payer, total, category, split method), SplitShare (per-member amount), Bill (due date, amount), Budget (monthly income, totals), Settlement/Payment (who→who, amount, status); keep common fields in a base schema to reduce duplication via Mongoose best practices.[2] [3]
- For extensibility (equal/unequal splits, different expense types), consider Mongoose discriminators or lightweight composition patterns to avoid schema bloat.[6] [7]

Navigation and layout

- Create a base layout.ejs (topbar, left menu, flash area), and partials for sidebar and header; include a "Due Bills" slot in the sidebar from the start so later features just feed data into it. [8] [9]

Authentication and roles

- Build pages: GET /auth/login, /auth/register; implement POST handlers; set up auth middleware and role checks (admin-only areas for reviewing split and exporting PDF).[5] [1]
- Add basic session/JWT guards to protect dashboard and group pages, enabling end-to-end flows during early tests.[10] [1]

Groups and members

- Pages: Groups List, Create Group; Group Detail with Members tab; Admin-only "Add member" modal (name, contact); server-side validation for phone/name.[11] [8]
- Routes: CRUD for groups/members; store members as embedded docs or separate collection with refs if reused across groups (choose now to avoid refactors).[12] [2]

Expense creation and splitting

- Pages: "Add Expense" in a group with fields: title, date, payer, total, participants, split type (equal/percentage/amounts); show live per-member share preview.[9] [11]
- Workflow: Save Expense draft → compute per-member SplitShares → admin review screen -> approve to lock and post balances; keep an audit trail for later settlements and PDFs.[8] [11]

Unequal split handling

- In the "Add Expense" flow, enable switching between equal, percent, and custom amounts with validation that sums match total; persist the chosen method in the model to support accurate recompute later.[3] [11]
- Admin review page: display all member shares, allow corrections before finalizing; finalized expenses should be immutable (create adjustments rather than edits).[9] [8]

Budget tracking

- Pages: Budget Settings per month (submit total income), and Monthly Overview (influx/deflux); connect to expenses posted in that month to compute totals.[8] [9]
- Routes: Create/Update monthly Budget; services to aggregate totals by month for dashboard cards.[2] [3]

Due bills module

- Sidebar "Due Bills" shows list of upcoming/overdue bills with amount and name; link to a Bills page with CRUD and status (due, paid), and optional reminders.[11] [9]
- Keep Bills separate from Expenses if bills are obligations that may or may not convert into expenses; otherwise, map paid bills to expenses on payment.[12] [2]

Settlements and balances

- Balancing service: from finalized expenses, compute net graph of who owes whom; show "Payments to make" and "Amount to be received" per user/group.[11] [8]
- Pages: Record Settlement (payer, receiver, amount, proof/ref); posting a settlement updates outstanding balances and dashboard figures.[13] [11]

Dashboard build-out

- Dashboard cards: Amount to receive, Payments to make, Bills to be paid, Count of bills; charts for monthly influx/deflux and category spend; aggregate queries with indexes for month and group.[9] [8]
- Start with static cards powered by services; add charts last, after aggregates are correct and performant.[14] [15]

PDF export

- After admin approves a split, enable "Download PDF" to export the split summary (expense details, member shares, approvals); add a group/month statement export later.[11] [9]
- Ensure exports reflect immutable records (finalized expenses + settlements) to avoid discrepancies on later edits.[8] [11]

Page-by-page build order

-
  1. Auth pages (login/register), base layout, and protected middleware; deploy a hello dashboard to validate pipeline.[1] [4]
-
  2. Groups list/detail and Members management (admin add member) to anchor other features to a group context.[8] [11]
-

3. Add Expense flow with equal split only, draft → finalize; show finalized list in group.[9] [11]

-

4. Expand split types to unequal (percentage/custom amounts) with validation and admin review.[3] [11]

-

5. Settlements page and balances computation; add “Payments to make/receive” cards to dashboard.[13] [8]

-

6. Budget module (monthly income), monthly totals (influx/deflux), and connect to dashboard.[9] [8]

-

7. Bills module with sidebar due list, bill detail, and mark as paid; link paid bills to expenses if applicable.[12] [11]

-

8. PDF export for splits/statements; polish dashboard charts and summaries.[14] [11]

Non-UI workflows

- Validation and security: server-side validation for amounts, membership, roles; store secrets in env; index Mongo fields used for lookups/aggregates.[1] [2]
- Aggregations: precompute or cache per-month and per-group aggregates to keep dashboards snappy, following Mongoose schema design best practices.[3] [12]

Milestone checkpoints

- Milestone A: Auth + Layout + Groups/Members complete; navigate end-to-end in dev.[1] [8]
- Milestone B: Expense add/review/finalize (equal split); balances list visible.[11] [9]
- Milestone C: Unequal split + settlements; export split PDF works.[8] [11]
- Milestone D: Budget + Bills + Dashboard cards; charts and polish.[14] [9]

What to postpone

- Postpone notifications, multi-currency, OCR, and advanced analytics until core flows stabilize; these commonly introduce scope creep and data model churn.[11] [8]

⁂

1. https://www.corbado.com/blog/nodejs-express-mongodb-jwt-authentication-roles
2. https://mongoosejs.com/docs/guide.html
3. https://www.freecodecamp.org/news/how-to-write-cleaner-code-using-mongoose-schemas/
4. https://learn.microsoft.com/en-us/azure/app-service/quickstart-nodejs
5. https://dev.to/dipakkr/implementing-authentication-in-nodejs-with-express-and-jwt-codelab-1-j5i

6. https://moldstud.com/articles/p-mongoose-schema-inheritance-best-practices
7. https://moldstud.com/articles/p-mastering-mongoose-schemas-best-management-practices
8. https://github.com/learner-enthusiast/expense-split-app-revamped
9. https://github.com/tuzup/SplitApp
10. https://www.loginradius.com/blog/engineering/guest-post/nodejs-authentication-guide
11. https://github.com/Harikrishnan-HK/Ease-Split
12. https://www.ayoubkhial.com/blog/mean-web-app-part-7-designing-and-modeling-database-with-mongoose
13. https://www.youtube.com/watch?v=H9M02of22z4
14. https://www.youtube.com/watch?v=yTh7LzEGpZ4
15. https://www.youtube.com/watch?v=PQnbtnsYUho
16. https://github.com/cvtqx/ExpenseSplitter
17. https://github.com/topics/expense-splitter?l=typescript
18. https://www.youtube.com/watch?v=ITNaVrDk9G0
19. https://www.linkedin.com/posts/arpit-jain-b7b940257_expensesplitter-grouppayments-innovation-activity-7142062151988412417-bYAf
20. https://labex.io/tutorials/javascript-building-a-modern-expense-splitter-web-app-298942