PRODUCT REQUIRED DOCUMENT

# TripSync by Piyush Funde

*Smart Group Travel Planner*

# 1. Discovery Insights

## 1.1. Who I Talked To

16 interviews over 4 days. Respondents spanned Pune, Bangalore, Mumbai, Delhi, Varanasi, and Boston (ages 23-40). I covered 6 active organizers, 4 passive travelers, 3 execution-willing contributors, and 3 mixed-role travelers. Trip contexts ranged from 3-person family road trips to 14-person all-women treks, a 10-person Bali trip, and a Ladakh trip stalled for 3+ years.

## 1.2. Six Key Findings

**1. The real problem is commitment, not planning.** Every organizer independently invented the same workaround: forcing commitment before planning begins. Akanksha collects Rs 5,000 non-refundable advances. Ramya runs a Zoom call where 30 interested people must commit on the spot (only 10 do). Gopal collects 80-90% of budget upfront. Three people, three solutions, one problem. The funnel from "interested" to "committed" loses two-thirds of participants.

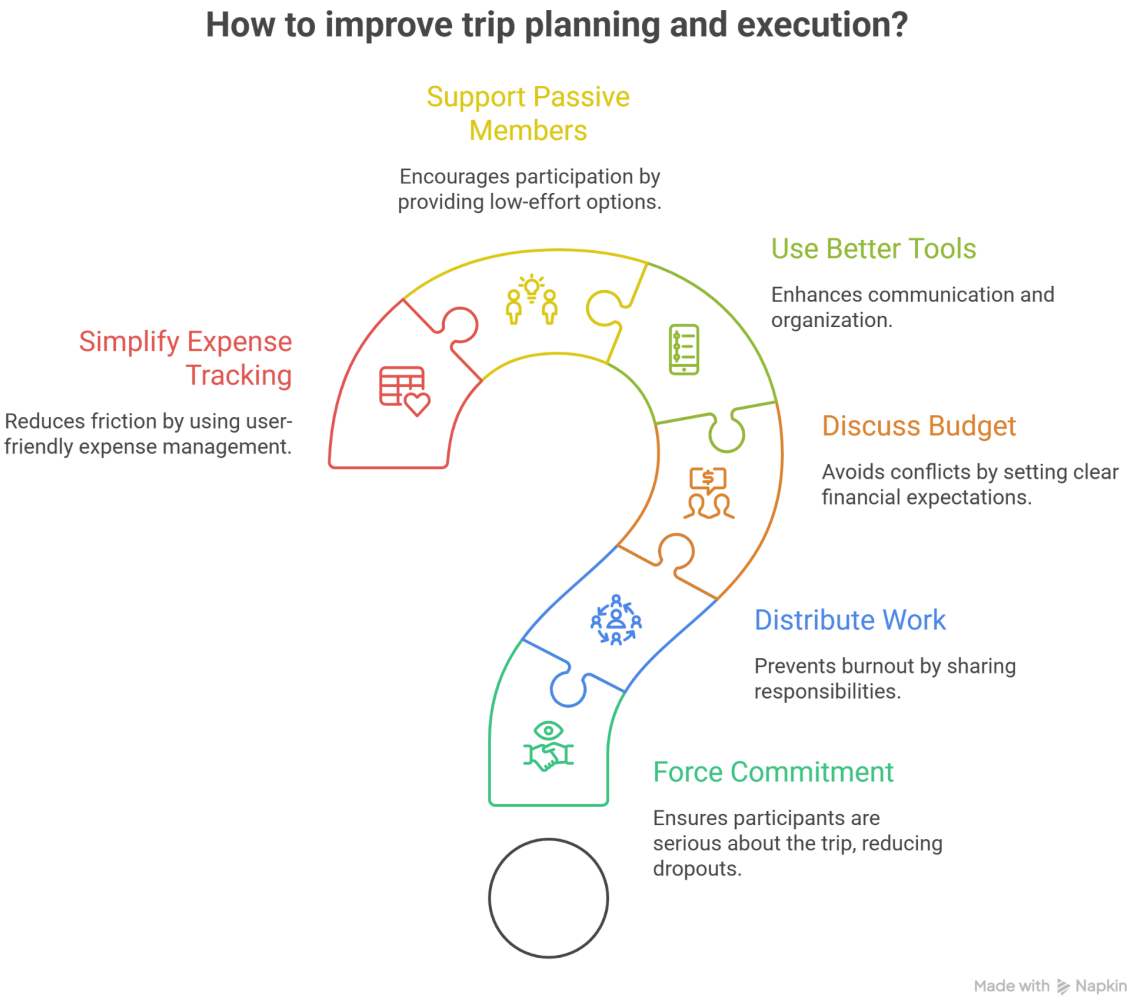
**2. One person absorbs all the work.** In 14/16 interviews, a single person handled everything: research, booking, budgeting, coordination, and conflict resolution. Akanksha: *"Maximum part came on me. Booking, traveler, common experiences, done by me."* Even organizers who enjoy planning get burned when delegated tasks fail. Pranav assigned someone to book Sagrada Familia tickets in Spain. They forgot. *"You had one task."*

**3. Budget is never discussed, then causes conflict.** Most groups avoid the budget conversation entirely. Akhil: *"We don't talk about budget. I don't know why."* Result: Akhil's group announced Rs 30K per person for Goa, spent Rs 60K+. In one case, an unspoken budget gap split a friend group in half at an expensive cafe. Nobody said "I can't afford this." Half just walked away.

**4. WhatsApp is everywhere but failing.** All 16 interviewees use WhatsApp for planning. All 16 described its failures: messages buried, polls unused or expired, links lost, no way to see what's decided vs what's pending. People use it because everyone is already there, not because it works.

**5. Passive members aren't lazy; tools are too demanding.** In groups of 6+, 2-3 members don't participate. It's not disinterest. Vibudh avoids proposing destinations because past choices were criticized: *"If someone says yaar achcha ne lagra, my whole trip experience becomes bad."* He'll book hotels and arrange transport, just won't pick the destination. No tool lets him contribute without owning the decision.

**6. Expense tracking creates friction everywhere.** Splitwise's free tier limits 3 transactions/day, which breaks on group trips. Gopal rejected it entirely for close friends: *"Adding Rs 100 expenses makes me angry. It makes things ugly."* His alternative: a central fund where everyone contributes upfront.




### 1.3. Four User Types

**Decisive Organizer** (Akanksha, Ramya, Pranav): Takes full ownership with personal workarounds. Pain: doing everything alone, no dropout protection.

**Willing Executor** (Vibudh, Sundaram): Handles logistics but refuses to make decisions due to blame risk. Needs clear tasks, not open-ended asks.

**Passive Follower** (Chaitanya, multiple described members): Goes along with the plan. Not disengaged, just overwhelmed by tool effort. Needs one-tap, 30-second interactions.

**Family Planner** (Shardul, Ravi): Plans for dependents with unique constraints (elder mobility, medical prep, shorter travel days). Bears all responsibility with zero tool support.

## 2. Problem Prioritization

## 2.1. Why Coordination, Not Booking

The travel industry has dozens of tools for booking (MakeMyTrip, Booking.com, Airbnb) and itinerary building (Wanderlog, TripIt). What no product solves is the coordination layer that sits between "let's go somewhere" and "everything is booked." Our 16 interviews confirmed this: not a single person struggled with finding flights or hotels. Every single person struggled with getting their group to commit, align on budget, make decisions together, and track shared expenses without friction.

## 2.2. The Five Problems I am Solving (Ranked by Evidence)

I identified 19 pain points from primary research and prioritized them on two dimensions: Severity (1-5, how painful this is for users) and Solvability (1-5, how addressable this is by a product). The priority score is the product of these two (max 25). Pain points are tiered as P0 Critical (21-25), P1 Significant (16-20), and P2 Latent (11-15), alongside frequency count showing how many of our 16 interviews surfaced each pain point. The five problems below all scored P0 Critical and form our product scope.

**Problem 1: Commitment Gap (12/16 interviews)** Groups lose two-thirds of members between "I'm interested" and "I've committed." This happens because there is no formal mechanism to capture commitment. People say yes on WhatsApp, then go silent. Organizers have no way to distinguish genuine intent from polite enthusiasm. Every organizer I spoke to invented their own workaround (advances, Zoom calls, central funds) because no product addresses this. Until commitment is locked, no downstream planning is reliable.

**Problem 2: Organizer Tax (14/16 interviews)** One person absorbs all cognitive, financial, and emotional load. They research destinations, build itineraries, make bookings, collect documents, track expenses, answer repeated questions, and absorb blame when things go wrong. This is the most universally confirmed pain point. The organizer often has the worst trip experience because they are project-managing instead of relaxing. Existing tools do not distribute this burden; they concentrate it further by giving the organizer more to manage.

**Problem 3: No Single Source of Truth (14/16 interviews)** Trip information is fragmented across WhatsApp messages, Google Docs, Splitwise, booking emails, Instagram saves, and the organizer's head. No group member can open one place and see: who's confirmed, what dates, what's the plan, how much will it cost, what needs my input. The organizer becomes the human integration layer, answering the same questions repeatedly.

**Problem 4: Expense Tracking Friction (13/16 interviews)** Splitwise has daily transaction limits, real-time logging feels burdensome ("takes away from the experience"), close friends find per-transaction tracking socially awkward, and multi-currency entries cause reconciliation chaos. Post-trip settlement drags for weeks. Some debts go unpaid, damaging relationships. Aishwarya on a friend who wouldn't settle: *"I am not going with him again."*

**Problem 5: Passive Participation (12/16 interviews)** Current tools demand too much effort from non-organizers: scrolling through hundreds of messages, processing ambiguous asks, making decisions with incomplete context. The irony is that passive members activate on Day 1 of the trip and overload the organizer with complaints about decisions they chose not

to influence. Ramya: *"People won't be onboarded when the WhatsApp group happens. They will be onboarded on Day 1. Then too many free suggestions."*


### 2.3. Why These Five, Not Others

I also identified significant pain points around geographic clustering of attractions, current conditions at destinations, visa anxiety, and family-specific travel needs. These are real but either affect fewer users, have partial workarounds (Google Maps, Reddit), or apply to narrower trip types. The five problems above are universal across every interview, every trip type (friends, family, international, domestic), and every user persona. They form the minimum scope a product must address to be genuinely useful.

Here is the link to the problem prioritisation sheet

## 3. Proposed Solution

### 3.1. TripSync: One Link to Plan Any Group Trip

TripSync is a mobile-first web app that replaces the scattered combination of WhatsApp + Google Sheets + Splitwise with a single shareable trip link. The organizer creates the trip, shares a link, and the group does everything from that one link: RSVP, vote on decisions, check the plan, track expenses, and settle up.

Every member signs in with Google (one tap) so their identity works across phone, laptop, and tablet. No separate app to install. No new account to create. No learning curve.

### 3.2. How It Works Across the Trip Lifecycle


**Getting Committed (solves Problem 1 + 5)** The organizer creates a trip in under 2 minutes: name, optional destination, RSVP deadline. They get a shareable link and drop it in the WhatsApp group. Each member taps the link, signs in with Google, and hits "I'm In," "Maybe," or "Can't Make It." The dashboard shows live status: who's confirmed, who's pending. As the deadline approaches, the system sends email reminders to non-responders automatically. The organizer can also tap "Remind via WhatsApp" to send a pre-crafted personal nudge. No chasing. No ambiguity. No assumed silence-as-consent.

**Aligning on Budget and Decisions (solves Problem 2 + 3 + 5)** Once the group is locked, three quick inputs are collected from every member, each with its own deadline. Budget: an anonymous slider for total comfortable spend including transport. The system shows the overlap: "Group sweet spot: Rs 18K-22K." Destination: structured polls where members vote by tapping options, not typing in a chaotic WhatsApp thread. Activities: lightweight preference capture (adventure / chill / culture / nightlife). Every decision is logged with status: Proposed, Voting, Decided, Booked. Any member opening the link at any point sees exactly what's settled and what still needs their input.

**Distributing the Work (solves Problem 2)** The organizer assigns specific tasks to specific people with deadlines: "Shruti: book hotel by March 15." "Piyush: arrange airport cab by March 18." Each task is visible on the dashboard with status (Assigned, Done, Overdue). The system sends email notifications on assignment and reminders before deadlines. The organizer's burden gets distributed across the group instead of sitting on one person's shoulders.

**Tracking Money Without the Awkwardness (solves Problem 4)** Two modes, because different groups have different social dynamics. Quick Split mode: any member adds an expense in 4 taps (amount, description, who paid, split among whom). No daily transaction limits. Running balance visible to everyone. Settlement summary at the end shows optimized transfers: "Piyush pays Pranav Rs 3,200." Group Fund mode (for close friends): everyone contributes upfront to a shared pool. One treasurer logs expenses against the fund. No per-transaction tracking. No petty accounting. End of trip: surplus is split back or deficit is collected.

**One Place for Everything (solves Problem 3)** The trip dashboard is the single source of truth. One link, always current: who's confirmed, dates, destination, itinerary, budget status, task checklist, expense balances, and a decision log. The organizer never answers "what's the plan?" again. A member who missed 50 WhatsApp messages taps the link and knows everything in 30 seconds. A complete activity log tracks every interaction (votes, expense edits, task completions) for transparency and accountability.

### 3.3. What Makes This Different from Existing Tools

WhatsApp has everyone but no structure. Splitwise tracks expenses but has no planning. Wanderlog builds itineraries but has no commitment mechanics or expense tracking. Google Sheets is customizable but not mobile-first and requires the organizer to build everything from scratch.

TripSync is the only product that owns the full coordination journey: from "who's in" through "what's the plan" to "who owes whom." It doesn't compete with booking platforms. It fills the gap between deciding to travel together and having everything booked.

Here is the link to the solution prioritisation matrix (see sheet 2)

## 4. Implementation Plan

### 4.1. Tech Stack

Next.js 14 (App Router) with TypeScript and Tailwind CSS for the frontend. Supabase for the database (PostgreSQL), authentication (Google OAuth), and realtime updates. Resend for transactional emails (nudges and summaries). Deployed on Vercel with auto-deploy from GitHub. This stack was chosen because it gives a live, deployed URL on free tiers with minimal configuration, meeting the case study requirement of "deployed and accessible."

### 4.2. Build Sequence

The build follows the natural trip lifecycle. Each phase delivers a usable increment.

**Phase 1: Foundation + Commitment Layer** Google OAuth sign-in via Supabase Auth. Trip creation form (name, destination, dates, RSVP deadline). Shareable trip link with unique

slug. Member join flow (tap link, sign in with Google, join trip). RSVP with In/Maybe/Out buttons and deadline countdown. Live status board showing who's confirmed, pending, or out. WhatsApp share button with pre-crafted invite message. This phase alone delivers the single most valuable feature identified in research: converting "interested" to "committed."

**Phase 2: Alignment + Decision Making** Budget alignment input (anonymous slider, group overlap visualization). Structured polls with deadline enforcement and single-vote-per-member (enforced via Google auth). Decision log with status tracking (Proposed, Voting, Decided, Booked). Email nudge system: automated reminders 24 hours before deadlines, organizer-triggered individual or bulk reminders. Nudge center showing all pending actions across the trip.

**Phase 3: Task Distribution + Itinerary** Task assignment with member, deadline, and status tracking (Assigned, Done, Overdue). Email notification on assignment and overdue reminders. Day-wise itinerary builder with category tags (travel, stay, food, activity, free time). Chill mode option for reunion-focused trips that don't need a structured itinerary.

**Phase 4: Expenses + Settlement** Quick Split mode: add expense in 4 taps (amount, description, who paid, split among). Running balance per member. Group Fund mode: upfront contributions, treasurer logs expenses against shared pool, fund balance dashboard. Settlement summary with optimized transfers. Complete activity log tracking every interaction (expense edits, deletions, votes, task completions) for transparency.

**Phase 5: Polish + Testing** Empty states for all tabs. Mobile responsiveness audit (375px primary). Booking confirmation aggregator (manual entry). Cross-browser testing. Multi-user simultaneous testing (3+ people using the same trip). Performance optimization.

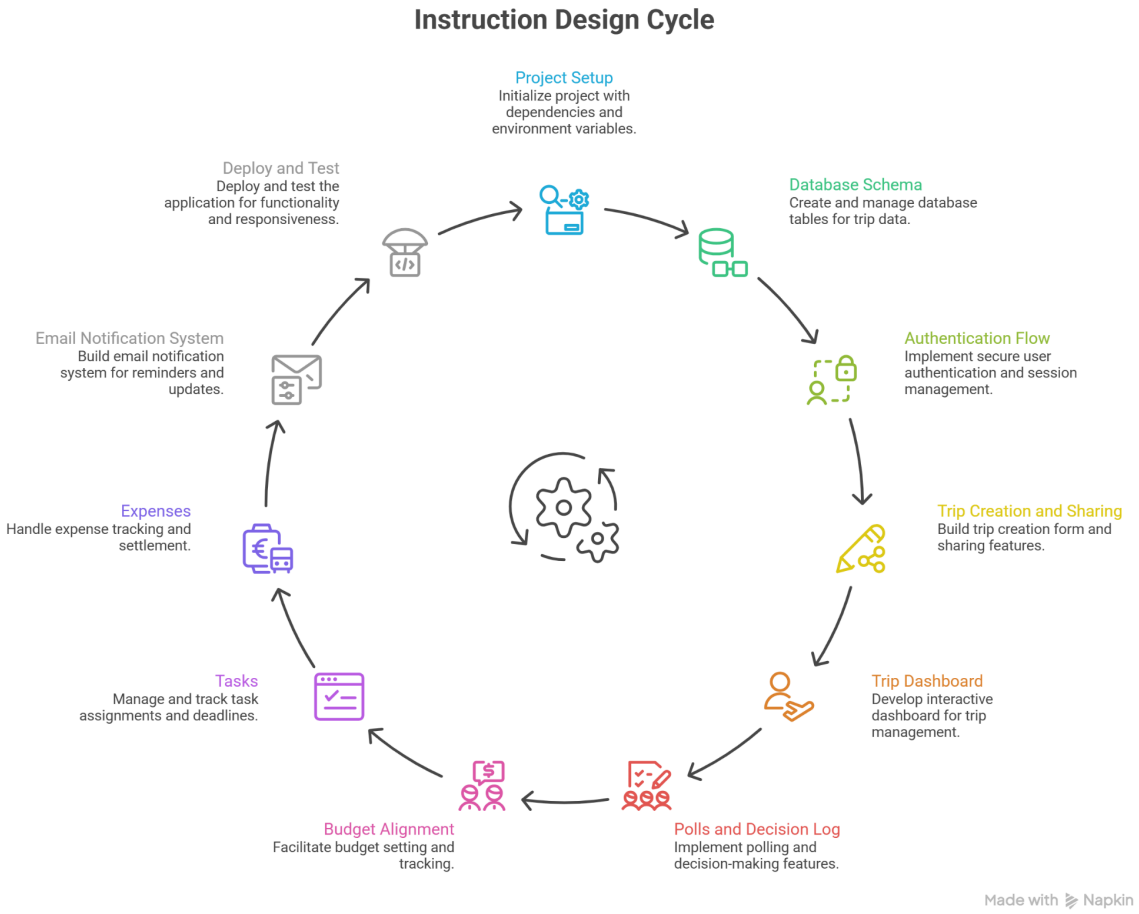
## 4.3. Database Architecture

12 core tables: trips, members (linked to Supabase Auth user IDs), polls, poll options, poll votes (unique constraint enforcing single vote), decisions, tasks, itinerary items, expenses, expense splits, group fund, fund contributions, fund expenses. Plus 2 system tables: activity log (complete audit trail of all interactions) and email notifications (log of all sent emails).

## 4.4. Deployment

GitHub repository connected to Vercel for automatic deployment on every push. Environment variables (Supabase URL, keys, Resend API key) configured in Vercel dashboard. Custom domain optional. The live URL is the submission deliverable.

# 5. Instruction Design: Step-by-Step Build Guide




## 5.1. Step 1: Project Setup

Initialize a Next.js 14 project with TypeScript, Tailwind CSS, and the App Router. Install dependencies: @supabase/supabase-js, @supabase/ssr, prisma, @prisma/client, swr, and resend. Create a Supabase project on supabase.com (free tier). Enable Google OAuth in the Supabase Authentication settings (this requires creating OAuth credentials in Google Cloud Console: create a project, enable the OAuth consent screen, add authorized redirect URI from Supabase dashboard, and copy the client ID and secret back into Supabase). Set up environment variables locally and in Vercel: NEXT_PUBLIC_SUPABASE_URL, NEXT_PUBLIC_SUPABASE_ANON_KEY, SUPABASE_SERVICE_ROLE_KEY, RESEND_API_KEY, and NEXT_PUBLIC_APP_URL.

## 5.2. Step 2: Database Schema

Run the full schema in the Supabase SQL editor. The schema creates 14 tables: trips (with slug, destination, dates, RSVP deadline, status), members (linked to Supabase Auth user_id with Google profile data: name, email, avatar_url, plus RSVP status and budget range), polls and poll_options for structured voting, poll_votes with a UNIQUE constraint on poll_id +

member_id to enforce single voting rights, decisions for logging group choices with status flow (Proposed to Voting to Decided to Booked), tasks with assignee, deadline, and status, itinerary_items organized by day_number and sort_order, expenses with amount/category/paid_by, expense_splits linking each expense to member shares, group_fund and fund_contributions and fund_expenses for the central fund mode, activity_log capturing every user interaction as a JSONB payload for the audit trail, and email_notifications logging all sent emails. After running the schema, initialize Prisma with npx prisma db pull and npx prisma generate to create the type-safe client.

### 5.3. Step 3: Authentication Flow

Create a Supabase auth helper using @supabase/ssr for server-side and client-side auth. Build a middleware.ts that checks the session on every request. The sign-in page shows a trip preview (name, organizer, member count) pulled from the public trip data, with a "Continue with Google" button below. On successful Google OAuth callback, check if the user_id exists in the members table for this trip. If yes, redirect to the dashboard. If no, show a "Join this trip?" confirmation (name and photo pre-filled from Google profile) and on confirmation insert a new member record. Store the session in cookies so it persists across page reloads. This gives cross-device continuity: same Google account on phone and laptop sees the same trip data.

### 5.4. Step 4: Trip Creation and Sharing

Build the /create page with a form: trip name (required), destination (optional), start/end dates (optional), and RSVP deadline (required, defaulting to 5 days out). On submit, generate a URL-friendly slug from the trip name plus a random 4-character suffix (e.g., goa-march-2026-x7k2). Insert the trip record and add the current user as the first member with role "organizer" and rsvp_status "in." Redirect to the dashboard with a share prompt showing the trip link. Include a "Copy Link" button using the Clipboard API and a "Share on WhatsApp" button that opens wa.me/ with a pre-filled message containing the trip name, link, and RSVP deadline.

### 5.5. Step 5: Trip Dashboard

Build /t/[slug]/page.tsx as the main dashboard with 5 tabs using client-side tab navigation (no page reloads). The Overview tab contains: trip info header (name, destination, dates, status badge, days-until-trip countdown), RSVP status board (list of all members with colored status badges, pending members showing WhatsApp and Email remind buttons for the organizer), budget snapshot (group range if collected, planned vs spent progress bar), quick stats (decisions made, tasks completed, open polls), and a recent activity feed (last 5 entries from activity_log). Use SWR for data fetching with revalidation so the dashboard updates when other members take actions. The tab bar sits at the bottom on mobile (fixed position) with 5 icons: Overview, Polls, Itinerary, Tasks, Expenses.

### 5.6. Step 6: Polls and Decision Log

The Polls tab shows active polls as cards with the question, options rendered as horizontal percentage bars, vote counts, the current user's vote highlighted, deadline countdown, and a

voter status line ("5/7 voted, not voted: Rahul, Meera"). The vote action is a POST to /api/trips/[slug]/polls/[id]/vote that checks the UNIQUE constraint. If the user has already voted, show their current selection with a "Change Vote" option (PUT replaces the previous vote). Poll creation (organizer only) is a modal with fields for question, options (add/remove dynamically), and deadline. When a poll closes (deadline passed), the system marks status as "closed" and auto-creates or updates the linked decision record with status "decided" and the winning option. Below the polls section, render the Decision Log as a list showing each decision's category icon, title, status badge (color-coded: grey for Proposed, blue for Voting, green for Decided, purple for Booked), and outcome text.

### 5.7. Step 7: Budget Alignment

Triggered from the Overview tab via a "Set up budget alignment" button (organizer). Creates a special poll-type interaction. Each member sees a dual-handle range slider (Rs 5,000 to Rs 1,00,000) to input their comfortable min and max budget. Values are stored in the member record's budget_min and budget_max fields. After all members submit (or deadline passes), the results view renders anonymous horizontal bars showing each range (no names) with the overlap zone highlighted in green and labeled "Sweet spot: Rs X to Rs Y." The trip record's budget_min and budget_max are updated with the overlap values.

### 5.8. Step 8: Tasks

The Tasks tab shows task cards with color-coded left borders (blue: assigned, amber: in progress, green: done, red: overdue). Each card displays: title, assignee (avatar + name), deadline with countdown, and status badge. The assigned member sees a "Mark as Done" button on their tasks. Task creation modal (accessible to organizer and any member): title, assign-to dropdown of trip members, deadline picker. On creation, insert into tasks table, log to activity_log, and send an email to the assignee via the Resend API with the task title, deadline, and trip link. A scheduled check (or on-page-load check) marks tasks as "overdue" when deadline has passed and status is not "done," triggering a reminder email to the assignee.

### 5.9. Step 9: Expenses

The Expenses tab has a mode toggle at the top: Quick Split (default) and Group Fund. In Quick Split mode, the "Add Expense" button opens a bottom sheet with: amount (large number input), description (text), category (tappable pills: Food, Transport, Stay, Activity, Shopping, Other), paid by (dropdown defaulting to current user), and split among (toggle between "Everyone" and "Select members" with checkboxes). On submit, insert into expenses table, create expense_splits records (equal division by default), and log to activity_log. The expense list shows entries chronologically with category icon, description, amount, payer, and split info. At the top, a balance summary card shows the current user's net position ("You owe Rs 3,200" or "Owed Rs 1,800"). A collapsible balance table shows all members' net positions. The Settlement screen calculates optimized transfers (minimum number of transactions to settle all debts) and shows payer-to-receiver cards with "Mark as Settled" buttons.

For Group Fund mode, the organizer sets target per person and designates a treasurer. The fund dashboard shows: total collected vs target (progress bar), per-member contribution status (paid/partial/unpaid with remind buttons), fund expense log (treasurer adds: description + amount + category), and remaining balance. End-of-trip: "Close Fund" calculates surplus (split back) or deficit (collect more).

Every expense action (add, edit, delete) is recorded in activity_log with before/after values in the JSONB detail field, creating a complete audit trail accessible via the Activity tab.

## 5.10. Step 10: Email Notification System

Build an email service using the Resend API. Create HTML email templates for 6 notification types: RSVP reminder (who's in/pending, deadline), poll reminder (what's open, who hasn't voted), task assignment (task title, deadline, link), task overdue (reminder to assignee, alert to organizer), trip summary (full status across all dimensions), and settlement summary (who owes whom, amounts). The organizer's nudge center (accessible via bell icon on dashboard) shows all pending actions grouped by type with "Send Email" and "WhatsApp Remind" buttons per person, plus bulk actions: "Email All Pending" and "Email Trip Summary to Everyone." Automated triggers fire via API route checks: 24 hours before any deadline, send reminder to members who haven't acted. When a poll closes, email results to all. When a task is assigned, email the assignee. All sent emails are logged in email_notifications with recipient, type, subject, and timestamp.

## 5.11. Step 11: Deploy and Test

Connect the GitHub repository to Vercel. Set environment variables in the Vercel dashboard. Push to main branch to trigger auto-deployment. Test the complete flow: create a trip on one device, open the link on a second device (different browser or incognito), sign in with a different Google account, join and RSVP. Verify: RSVP status updates in real time on the organizer's dashboard, polls enforce single vote per Google account, expenses calculate correct balances, tasks show correct assignment and status, email notifications are received. Test with 3+ simultaneous users to confirm the case study requirement. Test on mobile (375px) to verify responsive layout.