Headroom - Product Requirements Document

Your Financial Breathing Space

Version: 2.0

Document Owner: Aturinda David

Last Updated: January 2026

Status: Final - Ready for Development

Document Version History

- **v1.0** - Initial PRD creation based on project proposal
- **v1.1** - Updated branding to Headroom with tagline "Your Financial Breathing Space"
- **v2.0** - Major update: Added Projects & Wallets architecture, Category-first navigation, Extended to 10-week timeline

---

Table of Contents

1. Executive Summary

Product Name: Headroom

Tagline: Your Financial Breathing Space

Product Type: Mobile-first Personal Budget Assistant

Target Launch: 10-week MVP development cycle

Product Owner: Aturinda David

Vision Statement

Headroom empowers individuals to spend confidently and live debt-free by making budgeting effortless, automatic, and always accessible. The app's core philosophy centers on "Free to Spend" - showing users at a glance exactly how much they can spend guilt-free after accounting for bills, savings goals, and commitments.

Problem Statement

Modern users struggle with personal finance management due to:

- Manual expense tracking being tedious and time-consuming
- Lack of reliable offline access to financial data
- Difficulty maintaining consistent budgeting habits
- Poor visibility into spending patterns and debt obligations
- Disconnected experience across multiple devices
- Complex financial contexts requiring separate organization (personal vs. business, household vs. individual)
- Multiple money storage methods (cash, mobile money, bank accounts) difficult to track simultaneously

Solution Overview

Headroom is an offline-first, mobile-optimized budgeting application that automates recurring transactions, provides real-time budget tracking, and maintains seamless cross-device synchronization. The app eliminates friction in money management through smart automation and intuitive design.

Core Innovation - The Three-Layer System:

1. **Projects:** Isolated financial contexts (Personal, Business, Family Pool, etc.)
2. **Wallets:** Actual money stores within projects (Cash, Mobile Money, Bank Account, etc.)
3. **Categories:** Spending organization and navigation (Food, Transport, Bills, etc.)

The central feature is the **"Free to Spend"** amount - a prominent, always-visible number showing exactly how much money is available for discretionary spending after all commitments (bills, debt payments, savings goals) are accounted for. This transforms budgeting from restrictive to empowering.

2. Product Goals & Success Metrics

Primary Goals

1. Reduce Friction in Daily Expense Management Through Automation

- Users spend less than 30 seconds logging a typical transaction
- Recurring transactions generate automatically without user intervention
- Default wallets and remembered project context minimize repetitive selections
- 80% of transactions logged with 4 taps or fewer

Traditional budgeting tools require too much manual work. Headroom automates recurring bills, remembers context, and uses smart defaults to make expense tracking effortless.

2. Enable Offline-First Functionality for Uninterrupted Access

- 100% of core features work without internet connection
- Zero data loss during offline sessions
- Sync completes successfully >99% of the time
- Users in areas with unreliable connectivity can fully use the app

Financial tracking shouldn't stop when the internet is unavailable. Offline-first design ensures users can always log transactions, view their budget, and make financial decisions.

3. Promote Financial Awareness Through Real-Time Insights and Alerts

- Users always know their "Free to Spend" amount for current period
- Budget alerts help users stay on track before overspending
- Category-based organization reveals spending patterns
- Project separation provides clarity across different financial contexts

Financial awareness leads to better decisions. Real-time visibility into spending, combined with intelligent nudges, helps users stay within budget and reach their goals.

4. Foster Debt Reduction by Making Obligations Visible and Trackable

- Users link repayment transactions to specific debts
- Progress visualization motivates continued debt reduction
- Projected payoff dates help users plan
- Total debt visibility across projects (when needed)

Debt feels overwhelming when it's abstract. Headroom makes debt concrete, trackable, and manageable, turning an emotional burden into a solvable problem.

5. Deliver Seamless Experience Across All User Devices

- Projects, wallets, and transactions sync across devices in <5 seconds
- Users can start on mobile and continue on desktop without friction
- Active project remembered across sessions and devices
- Consistent UI patterns across platforms

Modern users switch between devices constantly. Headroom should feel like one continuous experience, not fragmented across platforms.

6. Provide Flexible Organization Through Projects and Wallets

- Users create multiple projects for different financial contexts
- Wallet-based tracking reflects real-world money distribution
- Project switching is instant and intuitive
- Financial data remains cleanly separated by context

Everyone's financial life is complex and multifaceted. Projects and wallets allow users to organize finances in ways that match their real-world situations (personal vs. business, cash vs. digital).

Success Metrics (Post-Launch)

User Engagement Metrics

Metric	Target	Measurement Period	Definition
User Retention Rate	>60%	30 days	Percentage of users who return after 30 days
Daily Active Users (DAU)	Growth of 20% MoM	Monthly	Unique users who open app daily
Weekly Active Users (WAU)	>70% of registered users	Weekly	Unique users active in past 7 days
Average Session Duration	>3 minutes	Weekly	Average time spent per app session
Sessions per User per Week	>5 sessions	Weekly	Average number of times user opens app

High retention and frequent usage indicate Headroom has become a habitual part of users' financial routines.

Feature Adoption Metrics

Metric	Target	Measurement Period	Definition
Transaction Logging Rate	>80% of users log ≥1 transaction/week	Weekly	Users actively tracking expenses
Recurring Transaction Setup	>40% of users	30 days	Users who set up at least one recurring transaction
Multiple Projects Usage	>25% of users	60 days	Users who create and use 2+ projects
Multiple Wallets Usage	>60% of users	30 days	Users who create 2+ wallets in any project
Category Budget Setup	>50% of users	30 days	Users who set budgets for at least one category

Debt Tracking Usage	>30% of users	60 days	Users who add at least one debt to track
Custom Category Creation	>20% of users	60 days	Users who create at least one custom category

Feature adoption indicates users are moving beyond basic expense logging to fuller budget management and organization.

Offline & Sync Performance Metrics

Metric	Target	Measurement Period	Definition
Offline Usage Rate	>40% of sessions	Monthly	Percentage of app sessions with no internet
Sync Success Rate	>99%	Daily	Percentage of sync operations that complete without error
Average Sync Time	<5 seconds	Daily	Time to sync pending transactions when online
Data Loss Rate	<0.01%	Monthly	Percentage of transactions lost due to sync failures
Offline Transaction Queue	Median <5 transactions	Weekly	Number of pending transactions waiting to sync

Offline-first is a core value proposition. These metrics ensure the offline experience works flawlessly.

Financial Behavior Metrics

Metric	Target	Measurement Period	Definition

Budget Goal Completion	>50% of users	Monthly	Users who stay under their overall monthly budget
Category Budget Adherence	>60% of categories	Monthly	Percentage of category budgets not exceeded
Free to Spend Accuracy	User-reported satisfaction >80%	Quarterly Survey	Users who trust Free to Spend calculation
Debt Reduction Tracking	>70% of debt-tracking users show progress	Monthly	Users with decreasing debt balances
Average Transactions per User	>20 per month	Monthly	Active financial tracking behavior

These metrics show Headroom is actually helping users manage their finances better, not just tracking data.

User Experience Metrics

Metric	Target	Measurement Period	Definition
Transaction Entry Time	<30 seconds	Weekly	Average time from FAB tap to transaction saved
Dashboard Load Time	<2 seconds	Daily	Time to render dashboard with Free to Spend
Project Switch Time	<1 second	Weekly	Time to switch between projects
App Crash Rate	<0.5%	Daily	Percentage of sessions ending in crash
API Error Rate	<1%	Daily	Percentage of API requests that fail
User-Reported Bugs	<5 per 100 users	Monthly	Bug reports via feedback channel

Speed and reliability directly impact user satisfaction and retention. Slow or buggy apps get abandoned.

Project & Wallet Organization Metrics

Metric	Target	Measurement Period	Definition
Average Projects per User	1.5-2.5	Monthly	Users typically maintain 1-3 active projects
Average Wallets per Project	2-3	Monthly	Most projects have 2-3 active wallets
Primary Project Usage	>80% of transactions	Monthly	Percentage in user's primary project
Project Switch Frequency	<2 times per session	Weekly	Users mostly stay in one project per session
Wallet Transfer Frequency	>10% of transactions	Monthly	Users actively moving money between wallets

These metrics help validate the project/wallet model is working as intended without overwhelming users.

Onboarding & Tutorial Metrics

Metric	Target	Measurement Period	Definition
Onboarding Completion Rate	>85%	Per cohort	Users who complete Phase 1 setup
Time to First Transaction	<10 minutes	Per user	Time from signup to first logged transaction
Tutorial Skip Rate	<30%	Monthly	Users who skip all tutorials
Tutorial Re-enable Rate	<5%	Monthly	Users who re-enable tutorials after skipping

Extended Setup Completion	>40%	Week 1	Users who complete delayed setup prompts

Good onboarding drives activation and long-term retention. High completion rates indicate frictionless setup.

Growth & Acquisition Metrics

Metric	Target	Measurement Period	Definition
Monthly Sign-ups	Growth of 25% MoM	Monthly	New user registrations
Activation Rate	>70%	Weekly	New users who log at least 3 transactions in first week
Referral Rate	>15%	Quarterly	Users who invite others (future feature)
Organic vs. Paid Ratio	>60% organic	Monthly	Source of new user acquisition

Sustainable growth requires both new user acquisition and strong activation, indicating product-market fit.

North Star Metric

Primary North Star Metric: Weekly Active Transactions - The number of transactions logged per week across all users

Supporting Metrics:

- Free to Spend accuracy (user trust)
- Budget adherence rate (behavioral change)
- Multi-project usage (organization value)
- Offline session rate (core value prop)

3. Target Audience

Primary Users

1. **Young Professionals (Age 22-35)**

   Demographics:

   - Early to mid-career individuals managing first regular salaries
   - Tech-savvy, comfortable with mobile apps
   - Often juggling multiple financial responsibilities (rent, student loans, savings)
   - May have side hustles or freelance income alongside main job

   Characteristics:

   - Need simple expense tracking without requiring financial expertise
   - Value mobile-first, on-the-go access for quick transaction logging
   - Often in areas with unreliable internet connectivity (commutes, cafes, rural areas)
   - Want to feel in control without spending hours on spreadsheets
   - Appreciate automation and smart features
   - Likely managing multiple financial contexts (personal + side business)

   Pain Points:

   - Traditional budgeting feels restrictive and tedious
   - Forget to log expenses, leading to inaccurate tracking
   - Lose track of subscriptions and recurring payments
   - Don't know how much they can safely spend without overspending
   - Struggle to separate business expenses from personal ones
   - Need to track both digital payments (cards, mobile money) and cash

   How Headroom Helps:

   - Projects separate personal from side business finances
   - Wallets track multiple payment methods (cash, mobile money, bank)
   - Quick transaction logging (<30 seconds) reduces friction
   - Recurring transactions automate bill tracking
   - Free to Spend gives instant spending clarity
   - Offline-first works on unreliable connections

2. **Budget-Conscious Individuals**

   Demographics**:**

- Age 25-45, diverse income levels
- Actively working to reduce debt or increase savings
- Previously struggled with traditional budgeting tools
- May have tried and abandoned spreadsheets or complex software
- Looking for sustainable, long-term financial management

Characteristics:

- Motivated to improve financial situation
- Willing to track expenses if tool is effortless
- Prefer visual feedback over numbers-heavy reports
- Value automation over manual data entry
- Need encouragement and positive reinforcement
- Want to see progress toward goals (debt reduction, savings)

Pain Points:

- Traditional budgeting tools feel like homework
- Difficult to stay motivated when tools are complex
- Hard to visualize debt reduction progress
- Guilt around spending, even when within budget
- Lose track of where money actually goes
- Can't organize finances by different priorities or contexts

How Headroom Helps:

- Free to Spend removes spending guilt (know what's safe to spend)
- Category-first organization reveals spending patterns clearly
- Debt tracking with visual progress motivates continued reduction
- Budget alerts prevent overspending before it happens
- Projects can separate "debt payoff" context from "daily life"
- Smart nudges provide encouragement, not judgment

User Personas

1. Persona 1: Sarah - The Busy Professional

Background:

- **Age:** 27
- **Occupation:** Marketing Manager
- **Location:** Kampala, Uganda
- **Monthly Income:** UGX 4,200,000 (~$1,120 USD)

- **Tech Comfort:** High - uses multiple apps daily, comfortable with mobile-first tools

Financial Situation:

- Renting apartment (UGX 800,000/month)
- Student loan: UGX 15,000,000 remaining
- Saving for wedding (goal: UGX 20,000,000 in 18 months)
- Has side hustle doing freelance social media consulting (UGX 500,000-1,000,000/month)
- Uses Mobile Money (primary), bank account, and some cash

Pain Points:

- Forgets to log expenses during busy workdays
- Loses track of freelance income vs. expenses
- Subscriptions (Netflix, Spotify, gym) auto-renew and surprise her
- Doesn't know if she can afford spontaneous purchases
- Hard to see if she's on track for wedding savings goal
- Internet unreliable during commute when she'd log transactions

Goals:

- Stay within monthly budget without feeling restricted
- Save UGX 1,100,000/month for wedding
- Keep freelance finances separate from personal
- Reduce student loan faster if possible
- Know instantly: "Can I afford this coffee/lunch/purchase?"

How Sarah Uses Headroom:

1) **Two Projects:**
    a) "Personal" - Main job income, rent, living expenses, student loan
    b) "Freelance" - Side hustle income and business expenses
2) **Multiple Wallets:**
    a) Personal: Mobile Money, Bank Account, Cash
    b) Freelance: Business Mobile Money, Business Bank
3) **Key Features Used:**
    a) Logs expenses offline during commute, syncs at office
    b) Recurring transactions for rent, loan payment, subscriptions
    c) Category budgets (Food: UGX 600,000, Entertainment: UGX 300,000)
    d) Checks Free to Spend before every purchase
    e) Tracks student loan repayments in debt tracker
4) **Usage Pattern:**
    a) Opens app 3-5 times daily for quick transaction logs
    b) Checks dashboard every morning
    c) Reviews weekly spending summary on Sundays

d) Switches to Freelance project when working on side hustle

Success Metrics for Sarah:

- Logs 90%+ of transactions (vs. 50% with previous tools)
- Stays under budget 85% of months
- Feels confident about spontaneous purchases
- Wedding savings on track

2. Persona 2: James - The Debt Reducer

Background:

- **Age:** 31
- **Occupation:** Software Developer
- **Location:** Nairobi, Kenya
- **Monthly Income:** KES 600,000 (~$4,650 USD)
- **Tech Comfort:** Very High - appreciates automation, API integrations, technical tools

Financial Situation:

- Student loans: KES 2,500,000 remaining (4 years left at current rate)
- Personal loan: KES 800,000 remaining
- Credit card debt: KES 150,000
- Wants to be debt-free in 2 years (aggressive payoff plan)
- Lives frugally to maximize debt payments
- Uses mainly mobile banking and credit card

Pain Points:

- Multiple loan repayments with different schedules confusing
- Unclear timeline for becoming debt-free
- Hard to visualize progress when paying off debt slowly
- Tempted to overspend when doesn't see clear impact
- Difficult to know how much extra to put toward debt each month
- Wants to celebrate wins but hard to see incremental progress

Goals:

- Pay off all debt in 2 years (needs to pay extra KES 80,000/month)
- Visualize debt reduction progress clearly
- Know exactly how much is "free" after debt obligations
- Track where money goes to find more to put toward debt

- Stay motivated through long payoff journey

How James Uses Headroom:

1. **Single Project:** "Personal" (may add "Emergency Fund" project later)
2. **Wallets:**
    - Bank Account (salary deposits here)
    - M-Pesa (daily spending)
    - Credit Card (minimizing use)
3. **Key Features Used:**
    - All three debts tracked in Debt Tracker with progress bars
    - Recurring transactions for all loan payments
    - Links repayment transactions to specific debts
    - Category budgets very tight (Food: KES 15,000, Transport: KES 5,000)
    - Free to Spend shows what's left after debt obligations
    - Weekly insights show if he's underspending (can put more to debt)
4. **Usage Pattern:**
    - Checks Free to Spend before every purchase
    - Reviews debt progress weekly
    - Celebrates when debt milestones hit (projected payoff date moves up)
    - Analyzes Reports monthly to find spending to cut
    - Adjusts budgets quarterly to maximize debt payments

Success Metrics for James:

- Debt balance decreases month-over-month
- Projected payoff date shortens from 4 years to 2 years
- Stays under tight budgets 90% of time
- Feels motivated and in control (not overwhelmed)

3. Persona 3: Amina - The Family Organizer

Background:

- **Age:** 34
- **Occupation:** High School Teacher
- **Location:** Dar es Salaam, Tanzania
- **Monthly Income:** TZS 1,800,000 (~$750 USD)
- **Tech Comfort:** Medium - uses WhatsApp, social media, basic apps

Financial Situation:

- Married with 2 children (ages 7 and 4)
- Household income: TZS 3,500,000 (combined with spouse)

- Manages household expenses
- Contributes to family savings pool with siblings (for parents' care)
- School fees, groceries, utilities split with spouse
- Uses mix of mobile money, bank account, and cash

Pain Points:

- Hard to track which expenses are personal vs. household vs. family pool
- Cash spending (market, transport) often forgotten
- Needs to account for money given to children/spouse
- School fees come quarterly and surprise her
- Difficult to show spouse where household money goes
- Family pool contributions sometimes missed

Goals:

- Keep household budget organized and transparent
- Track personal spending separately from joint expenses
- Remember quarterly school fees and large expenses
- Show progress on family savings pool
- Ensure kids' allowances tracked properly

How Amina Uses Headroom:

1. **Three Projects:**
    - "Personal" - Her own money (salary, personal expenses)
    - "Household" - Joint expenses with spouse (rent, utilities, groceries, school fees)
    - "Family Pool" - Contributions to parents' care fund with siblings
2. **Wallets Per Project:**
    - Personal: Mobile Money, Cash
    - Household: Joint Bank Account, Cash
    - Family Pool: Family Savings Account
3. **Key Features Used:**
    - Separate budgets per project (avoids mixing contexts)
    - Recurring transactions for rent, utilities, school fees (quarterly)
    - Category budgets help track household spending (Food, Education, Healthcare)
    - Transfers between wallets when moving money between contexts
    - Free to Spend per project shows what's available in each context
4. **Usage Pattern:**
    - Logs household expenses during evening (offline at market during day)
    - Checks Household project before big purchases
    - Reviews personal vs. household spending monthly
    - Switches to Family Pool project when contributing

    - Shows spouse Reports to discuss household budget

Success Metrics for Amina:

- All three financial contexts cleanly separated
- No more "Where did the money go?" questions from spouse
- School fees never surprise her (recurring reminders)
- Family pool contributions consistent
- Feels organized and in control across multiple roles

Secondary Audiences

1. Small Business Owners

Characteristics:

- Need to separate business from personal finances
- Track expenses for tax purposes
- Multiple income streams
- May need team collaboration features (future)

How Headroom Could Help:

- Business project separate from personal
- Category budgets for business expense types
- Export for tax preparation
- Future: Team access, receipt scanning

2. Students & Recent Graduates

Characteristics:

- Limited income (part-time work, allowances)
- Learning financial management for first time
- Very price-sensitive
- Need simple, intuitive tools

How Headroom Could Help:

- Free tier with all core features
- Simple interface, gentle learning curve
- Debt tracking for student loans

- Projects for "School" vs. "Personal"

3. Freelancers & Gig Workers

Characteristics:

- Irregular income
- Multiple clients/income sources
- Need to track business expenses
- Self-employed tax obligations

How Headroom Could Help:

- Projects for each major client or "Business" vs. "Personal"
- Category tracking for deductible expenses
- Income tracking per project
- Export for tax filing

4. Core Features (MVP)

4.0 Quick Sign-In with OAuth

Priority: P0 (Must-Have)

Description: Frictionless authentication using OAuth providers (Google, Apple, Microsoft) to minimize setup time and reduce barriers to entry. Users can sign in with a single tap, with optional email/password as backup.

User Stories:

- As a user, I want to sign in with my Google account so I can start using the app immediately
- As a user, I want my sign-in to work across all my devices without re-entering credentials
- As a user, I don't want to remember another password

Acceptance Criteria:

- One-tap sign-in with Google OAuth
- Apple Sign-In for iOS users
- Optional Microsoft OAuth for additional coverage
- Email/password as fallback option
- "Remember me" functionality with secure token storage
- Account creation and login in <5 seconds
- Seamless cross-device authentication
- Default "Personal" project automatically created on first sign-up

Technical Requirements:

- Supabase Auth with OAuth providers configured
- JWT token management with secure storage
- OAuth callback handling
- Session persistence across app restarts
- Automatic default project creation in database on new user registration

4.1 Projects & Financial Contexts

Priority: P0 (Must-Have)

Description: Projects are isolated financial environments that allow users to organize their finances into separate contexts (e.g., "Personal", "Side Business", "Family Pool", "Savings

Goal"). Each project has its own wallets, categories, budgets, debts, and transactions. Users interact with one project at a time, with the ability to switch contexts seamlessly.

User Stories:

- As a user, I want to separate my personal expenses from my side business finances so they don't mix
- As a user, I want to manage a family savings pool as a separate project with its own tracking
- As a user, I want to switch between projects easily without losing context
- As a user, I want to set one project as my primary/default that loads when I open the app
- As a user, I want to create, rename, and delete projects as my needs change

Acceptance Criteria:

- Users can create multiple projects with custom names
- Default "Personal" project created automatically on sign-up
- Only one project active at a time (all views show data for active project only)
- Project switcher accessible via dropdown in header showing current project name
- Users can set any project as "primary" (loads on app open)
- Projects can be edited (rename) or deleted (with confirmation)
- All financial data (wallets, categories, budgets, debts, transactions) is project-scoped
- Switching projects updates entire UI to show selected project's data
- Last active project is remembered across sessions (unless user has set a primary)
- Visual indicator clearly shows which project is currently active

Technical Requirements:

- Project data model with user relationship
- Project context state management (Redux/Context API)
- Project switching logic that clears and reloads all financial data
- Cascade delete handling when project is removed
- Default project creation on user registration
- Primary project flag in database

4.2 Wallets & Money Stores

Priority: P0 (Must-Have)

Description: Wallets are the actual money stores within a project (e.g., Cash, Mobile Money, Bank Account, Credit Card). They track balances and are the source/destination for all income and expense transactions. Wallets exist at the project level and do not assign purpose to money - that's the role of categories. Users can transfer money between wallets with optional fees.

User Stories:

- As a user, I want to track separate balances for my cash, mobile money, and bank account
- As a user, I want to see which wallet I'm spending from for each transaction
- As a user, I want to transfer money between wallets (e.g., withdraw cash from bank)
- As a user, I want to record transaction fees when transferring between wallets
- As a user, I want to set initial wallet balances and have them update automatically with transactions
- As a user, I want to manually adjust wallet balances to fix discrepancies with real-world balances

Acceptance Criteria:

- Users can create multiple wallets per project with custom names
- Common wallet types suggested during creation: Cash, Mobile Money, Bank Account, Credit Card
- Each wallet displays current balance
- Initial wallet balance can be set during creation
- Wallet balances update automatically based on transactions (income adds, expense subtracts)
- Users can manually edit wallet balance at any time (generates adjustment transaction)
- Transfer transaction type moves money between wallets (single transaction, not two)
- Transfers can include optional fee (recorded as metadata on transfer transaction)
- Transfer fees reduce the destination wallet balance appropriately
- All income/expense transactions must specify a wallet
- Dashboard shows wallet balance cards for quick overview
- Default wallet can be set per project for faster transaction entry
- Wallets can be archived (hidden but data retained) or deleted

Technical Requirements:

- Wallet data model with project relationship
- Balance calculation engine (sum of all transactions for wallet)
- Transfer transaction type with source/destination wallet fields
- Transaction fee metadata field
- Balance adjustment transaction type for manual corrections
- Default wallet flag per project
- Wallet archive/active status

4.3 Categories as Organization & Navigation

Priority: P0 (Must-Have)

Description: Categories (Food, Transport, Shopping, Bills, Entertainment, Health, Other) have evolved from simple transaction tags into primary organization and navigation tools. Each category serves as a financial "bucket" with its own budget tracking, transaction history, and insights. Users navigate by category to see spending patterns and manage category-specific budgets. Categories exist within each project.

User Stories:

- As a user, I want to navigate to "Food" and see all my food-related transactions and spending
- As a user, I want to set a monthly budget for each category (e.g., $500 for Food)
- As a user, I want to add custom categories specific to my project (e.g., "Pet Care" for personal, "Marketing" for business)
- As a user, I want to remove standard categories I don't use to keep my interface clean
- As a user, I want to see category-level insights (spending trends, budget progress, biggest expenses)

Acceptance Criteria:

- Standard categories available in every new project: Food, Transport, Shopping, Bills, Entertainment, Health, Other
- Users can add unlimited custom categories per project
- Users can remove/hide unused standard categories (suggested after 1 week if no transactions recorded)
- Each category displays: total spent this period, budget goal (if set), progress bar, recent transactions
- Categories serve as primary navigation on main screen (swipe or tap to enter category view)
- Within category view: list of all transactions for that category, add transaction button (pre-filled with category), budget settings
- Category budgets independent from overall project budget
- Visual indicators show budget status: green (under), yellow (approaching), red (exceeded)
- Categories can be reordered by user preference
- Category-level analytics: spending over time, average per transaction, largest expense

Technical Requirements:

- Category data model with project relationship
- Standard category seeding on project creation
- Custom category creation/deletion
- Category budget tracking linked to transactions
- Category-based transaction filtering and aggregation
- Category visibility/archive flag
- Category ordering/priority field

4.4 Smart Recurring Transactions

Priority: P0 (Must-Have)

Description: Recurring transactions are treated as an attribute rather than a separate entity. When adding any transaction, users can toggle "Make this recurring" and configure the frequency inline. This eliminates the need for a separate recurring transactions management flow. Recurring transactions are project-scoped and automatically generate at specified intervals.

User Stories:

- As a user, I want to mark my salary as recurring when I first add it, without going to a separate section
- As a user, I want my rent payment to auto-generate on the 1st of each month with an optional confirmation prompt
- As a user, I want to edit or pause a recurring transaction from the transactions list
- As a user, I want recurring transactions to respect the project and wallet they were created in

Acceptance Criteria:

- "Recurring" toggle available during any transaction creation
- When toggled on, additional fields appear inline: frequency, start date, optional end date, confirmation preference
- Recurring transactions appear in transaction lists with a recurring icon/badge
- System generates transactions automatically at specified intervals
- Generated transactions inherit: amount, category, wallet, project from original pattern
- Generated transactions work offline with local timestamp
- Users can edit, pause, or delete recurring patterns from transaction details
- Visual indicator distinguishes auto-generated from manual entries
- Editing recurring pattern offers: "This occurrence only" or "All future occurrences"
- Recurring transactions are scoped to their parent project

Technical Requirements:

- Unified transaction data model with optional recurring fields
- Local scheduling using IndexedDB timestamps
- Background task to check and generate due transactions
- Conflict resolution if transaction generated offline and synced later
- Project-scoped recurring logic

4.5 Transaction Logging with Wallets

Priority: P0 (Must-Have)

Description: All financial records (income, expense, transfers) are stored locally first (IndexedDB) and automatically synced to the cloud once connectivity is restored. Every transaction must specify a wallet (source for expenses, destination for income). The logging flow is conversational and sequential, minimizing cognitive load.

User Stories:

- As a user, I want to log expenses even without internet so I never miss recording a transaction
- As a user, I want to quickly select which wallet I'm paying from
- As a user, I want my wallet balance to update immediately after logging a transaction
- As a user, I want my offline entries to automatically sync when I'm back online
- As a user, I want visual feedback showing sync status

Acceptance Criteria:

- All transactions save locally immediately
- App functions fully without internet connection
- Transaction types: Income, Expense, Transfer
- Every transaction requires: type, amount, category, wallet (and destination wallet for transfers)
- Clear visual indicators show sync status (synced, pending, syncing)
- Automatic background sync when connection restored
- Manual "Sync Now" button for user control
- Conflict resolution handles duplicate entries
- Wallet balance updates in real-time as transactions are logged
- Transfer transactions handled as single transaction with source/destination

Technical Requirements:

- IndexedDB via Dexie.js for local storage
- Service Workers for background sync
- Queue system for pending transactions
- Timestamp-based conflict resolution
- Network status detection
- Wallet balance recalculation on transaction CRUD
- Transfer transaction logic

4.6 Budget Goals & Alerts

Priority: P0 (Must-Have)

Description: Users define category-specific spending limits within a project. Headroom visually tracks progress and provides alerts as users approach or exceed limits. Budget goals are project-scoped and can be set at both the project level (overall) and category level (specific).

User Stories:

- As a user, I want to set a monthly budget of $2,000 for my Personal project so I can control overall spending
- As a user, I want category budgets (e.g., $500 for Food) to track specific spending areas
- As a user, I want alerts when I reach 80% and 100% of any budget
- As a user, I want visual progress bars showing remaining budget in each category

Acceptance Criteria:

- Users can set overall project-level monthly budget
- Users can create category-specific budgets per project
- Real-time progress tracking with visual indicators (progress bars, percentage)
- Alerts at 80%, 100%, and 120% of budget limits
- Color-coded status (green = safe, yellow = warning, red = exceeded)
- Historical budget performance tracking per project
- Budget settings accessible from category views
- "Free to Spend" calculation factors in all category budgets and project budget

Technical Requirements:

- Real-time calculation engine for budget tracking (project-scoped)
- Alert system with configurable thresholds
- Category-budget linkage
- Notification system integration
- Budget history tracking per project

4.7 Debt Tracking

Priority: P1 (Should-Have)

Description: Users can record loans, repayments, and outstanding balances to monitor debt reduction progress within a project. Debts are project-scoped, allowing separation of personal debts from business debts or other contexts.

User Stories:

- As a user, I want to log my student loan ($50,000) in my Personal project and track repayments
- As a user, I want to see total outstanding debt for the current project
- As a user, I want to visualize my debt reduction progress over time
- As a user, I want to link repayment transactions to specific debts

Acceptance Criteria:

- Users can create debt records with: name, total amount, interest rate (optional), minimum payment
- Track multiple debts simultaneously per project
- Dashboard shows total outstanding debt for active project
- Visual progress bars for each debt
- Link repayment transactions (from specific wallet) to reduce debt balances
- Calculate projected payoff dates based on payment patterns
- Debts isolated per project

Technical Requirements:

- Debt data model with project relationship
- Debt calculation engine (project-scoped)
- Transaction linking system
- Progress visualization component
- Interest calculation (optional for MVP)

4.8 Mobile-First Dashboard

Priority: P0 (Must-Have)

Description: A card-based, responsive interface optimized for small screens, with the "Free to Spend" amount as the hero element. The dashboard shows data for the currently active project, with the project name displayed in the header. Wallet balance cards provide a quick overview of money distribution across wallets.

User Stories:

- As a user, I want to immediately see my "Free to Spend" amount for the active project when I open the app
- As a user, I want to see which project I'm currently viewing in the header
- As a user, I want to see my wallet balances at a glance
- As a user, I want to choose if I track my free spending daily, weekly, or monthly
- As a user, I want to navigate to categories quickly to add transactions or view spending
- As a user, I want to switch projects via the header dropdown

Acceptance Criteria:

- Dashboard loads in <2 seconds
- Header displays: current project name (clickable dropdown), sync status, settings icon
- "Free to Spend" amount is the largest, most prominent element (calculated for active project only)
- Formula: *(Total wallet balances) - (Committed expenses/bills for this period)*
- User selects period during onboarding: Daily, Weekly, or Monthly (default: Monthly)
- Rollover logic: Unspent Free to Spend carries to next period automatically
- Wallet balance cards show: wallet name, current balance, icon
- Category navigation: swipe or tap to view category-specific screens
- Shows: Free to Spend (with period indicator), wallet balances, top 3 spending categories
- Radial FAB (bottom right) with 3 actions: Dashboard, Categories (replaced Transactions), Debt Tracker
- Default FAB tap = Add Transaction; Long press = radial menu appears
- Card-based layout with smooth transitions
- Fully responsive (320px to 1440px+ screens)
- Touch-optimized interactions (swipe, tap, hold)

Technical Requirements:

- React components with Tailwind CSS
- Framer Motion for animations and swipe gestures
- Period-based calculation engine with rollover logic (project-scoped)
- Project context state management
- Wallet balance aggregation
- Touch gesture library (react-swipeable or similar)
- Radial menu component for FAB
- Lazy loading for performance

4.9 Cross-Device Sync

Priority: P0 (Must-Have)

Description: Data automatically synchronizes across multiple devices using the same account, maintaining consistent experience whether on phone, tablet, or web. All projects, wallets, categories, and transactions sync seamlessly.

User Stories:

- As a user, I want to log expenses on my phone and see them on my laptop
- As a user, I want my project settings and wallet balances to sync across all devices
- As a user, I want real-time updates when I make changes on any device

- As a user, I want to switch projects on one device and have it reflect on others

Acceptance Criteria:

- Same account accessible on multiple devices simultaneously
- Changes sync within 5 seconds when online
- Conflict resolution prioritizes most recent change
- Sync status visible on all devices
- Offline changes queue and sync when back online
- All project data (wallets, categories, budgets, debts, transactions) syncs completely

Technical Requirements:

- JWT-based authentication
- WebSocket or polling for real-time updates
- Conflict resolution algorithm (last-write-wins with timestamp)
- Cloud database (PostgreSQL via Supabase)
- Project-scoped sync logic

4.10 Smart Sync & Backup

Priority: P0 (Must-Have)

Description: Background sync layer ensures all offline transactions upload when online. Manual "Sync Now" option gives users direct control over synchronization across all projects.

User Stories:

- As a user, I want offline changes to automatically sync without my intervention
- As a user, I want to manually trigger sync to ensure all my projects' data is current
- As a user, I want to know if sync fails and why

Acceptance Criteria:

- Automatic background sync when connection detected
- Manual "Sync Now" button in settings/dashboard
- Sync status indicators (last synced timestamp)
- Error handling with user-friendly messages
- Retry logic for failed syncs
- Data integrity verification across all projects

Technical Requirements:

- Service Worker background sync

- Retry queue with exponential backoff
- Sync status state management
- Error logging and user notifications
- Project-aware sync operations

4.11 Reports & Time-Based Transaction Views

Priority: P1 (Should-Have)

Description: Since categories have become the primary navigation tool, traditional time-based transaction lists are moved to a "Reports" section. Users can access chronological transaction history, summaries, and simple statistics for the active project.

User Stories:

- As a user, I want to see all transactions for the current project in chronological order
- As a user, I want to filter transactions by date range
- As a user, I want to see my biggest expenses for the month
- As a user, I want simple stats like total income vs expenses for a period

Acceptance Criteria:

- "Reports" accessible from main navigation or settings
- Time-based transaction list showing all transactions for active project
- Date range filtering (this week, this month, custom range)
- Summary cards: total income, total expenses, net change for period
- "Biggest expense" and "most frequent category" highlights
- Searchable and filterable by category, wallet, amount range
- Export option (CSV) for transactions

Technical Requirements:

- Transaction aggregation and sorting by timestamp
- Date range filtering logic
- Summary calculation engine (project-scoped)
- Search and filter components
- CSV export functionality

4.12 Insights & Smart Nudges

Priority: P1 (Should-Have)

Description: Gentle notifications provide spending insights and encouragement, helping users stay aware of their financial behavior within each project without being intrusive. Notifications are intelligently timed for early morning (7-8 AM) and evening (6-8 PM) when users can actually review and act on them.

User Stories:

- As a user, I want encouraging messages when I'm under budget in my active project
- As a user, I want insights about my spending patterns within a project (e.g., "You spend most on weekends in your Personal project")
- As a user, I want helpful nudges at times when I can actually act on them, not random times
- As a user, I want notifications that respect my schedule and attention

Acceptance Criteria:

- Weekly spending summary notifications for active/primary project (scheduled for Sunday evening)
- Daily "Free to Spend" update for active/primary project (morning reminder)
- Positive reinforcement messages ("10% under Food budget in Personal project!")
- Spending pattern insights per project (time of day, day of week, category trends)
- Smart timing: Morning (7-8 AM) for daily updates, Evening (6-8 PM) for reviews
- Configurable notification preferences with time customization
- Maximum 2 notifications per day (excluding critical alerts)
- Users can disable specific insight types
- Quiet hours option (e.g., no notifications 10 PM - 7 AM)
- Project-specific insights

Technical Requirements:

- Firebase Cloud Messaging or similar
- Analytics engine for pattern detection (project-scoped)
- Notification scheduling system with time-based triggers
- User preference storage
- Timezone-aware scheduling
- Project context in notifications

5. Future Features (Post-MVP)

Phase 2 Features (Months 2-6 Post-Launch)

2.1 Predictive Suggestions & Smart Automation

Priority: High

Complexity: Medium

User Value: High

Description: AI/ML learns from user behavior to suggest likely transactions, auto-categorize expenses, and forecast monthly spending patterns.

Features:

- **Smart Transaction Suggestions:**
  - "You usually buy coffee at this time - add $5 to Food?"
  - Suggests based on location, time of day, historical patterns
  - One-tap to confirm and log transaction
- **Auto-Categorization:**
  - Learns from past categorization choices
  - Suggests category when user enters amount
  - "This $50 is usually Transport - confirm?"
  - Improves accuracy over time
- **Spending Forecasts:**
  - "Based on your patterns, you'll spend ~$450 on Food this month"
  - Early warnings: "You're spending faster than usual in Entertainment"
  - End-of-month projections: "At this rate, you'll finish $100 under budget"
- **Recurring Detection:**
  - Identifies repeated transactions: "This is your 3rd monthly $1,200 rent payment - make it recurring?"
  - Suggests making manual transactions automatic

Technical Requirements:

- Basic ML model (pattern recognition)
- Historical transaction analysis
- User feedback loop for training
- Privacy-preserving (on-device or anonymized)

Success Metrics:

- 60%+ of suggested transactions accepted

- Auto-categorization accuracy >80%
- Forecast accuracy within 15% of actual spending

2.2 Voice & Natural Language Input

Priority: Medium

Complexity: Medium

User Value: High

Description: Quick transaction logging via voice commands or natural language text input, making expense tracking even more effortless.

Features:

- **Voice Commands:**
  - "Add $25 for lunch" → Creates expense transaction
  - "I spent 50,000 shillings on groceries" → Parses amount and category
  - "Transfer $100 from bank to cash" → Creates transfer
  - Works offline, processes when back online
- **Chat-Style Input:**
  - Natural language text field: "Paid 15k for taxi"
  - AI parses: Amount: 15,000, Category: Transport, Type: Expense
  - User confirms and saves
- **Smart Parsing:**
  - Understands variations: "grabbed coffee - $5", "5 dollar coffee", "coffee 5"
  - Detects currency symbols and amounts
  - Infers category from keywords (coffee → Food, uber → Transport)
  - Asks clarifying questions if ambiguous

Technical Requirements:

- Speech-to-text API (Web Speech API or Google Cloud Speech)
- Natural language processing (simple regex or GPT-style model)
- Entity extraction (amount, category, wallet)
- Fallback to manual entry if parsing fails

User Experience:

- Microphone button on dashboard or in transaction flow
- Real-time transcription display
- Confirmation screen before saving
- Edit if parsing incorrect

Success Metrics:

- 40%+ of users try voice input
- Parsing accuracy >85%
- Average time to log transaction reduces to <15 seconds

2.3 Bank Statement Import & Parsing

Priority: High

Complexity: High

User Value: Very High

Description: Upload bank/mobile money statements (CSV, PDF, Excel) and automatically import transactions, reducing manual data entry.

Features:

- **Multi-Format Support:**
  - CSV from online banking
  - Excel/XLSX exports
  - PDF statements (OCR extraction)
  - Mobile money transaction SMS messages
- **Intelligent Parsing:**
  - Detects date, amount, description, balance columns
  - Handles different bank formats
  - Matches to categories based on merchant names
  - Identifies income vs. expenses
- **Duplicate Prevention:**
  - Compares to existing transactions
  - Flags potential duplicates for user review
  - "Found 23 new transactions, 5 possible duplicates"
- **Bulk Import Flow:**
  - Upload file or forward SMS
  - System parses and previews transactions
  - User reviews: categorize, edit, exclude duplicates
  - Confirm import
  - Transactions added to appropriate wallet
- **Bank/Provider Templates:**
  - Pre-configured parsers for common banks (Equity Bank, KCB, MTN Mobile Money, etc.)
  - User can create custom parsing rules
  - Community-contributed templates

Technical Requirements:

- CSV/Excel parsing library (Papaparse, SheetJS)
- PDF text extraction (pdf.js or external API)
- SMS parsing (for mobile money)
- Fuzzy matching for duplicate detection
- Transaction categorization engine

Success Metrics:

- 50%+ of users import at least one statement
- Import accuracy >90%
- Time to reconcile monthly transactions reduces by 80%

Privacy Considerations:

- Files processed locally when possible
- Server-side processing encrypted
- Files deleted after import
- User controls what data is uploaded

2.4 Enhanced Analytics Dashboard

Priority: Medium

Complexity: Medium

User Value: Medium-High

Description: Rich visualizations and deeper insights into spending patterns, trends, and financial health.

Features:

- **Trend Charts:**
  - Spending over time (line/bar charts)
  - Category breakdown (pie/donut charts)
  - Income vs. Expenses (stacked area chart)
  - Month-over-month comparisons
- **Comparative Analytics:**
  - "You spent 20% more on Food this month than last month"
  - Year-over-year comparisons
  - Project-to-project comparisons (if user has multiple)
- **Heatmaps:**

        - Spending by day of week
        - Spending by time of day
        - "You spend most on Friday evenings"
    - **Financial Health Score:**
        - Overall score (0-100) based on:
            - Budget adherence
            - Savings rate
            - Debt reduction progress
            - Spending consistency
        - Tips to improve score
    - **Goal Tracking:**
        - Savings goals with progress visualization
        - "You're 60% toward your $10,000 emergency fund"
        - Projected completion dates
    - **Merchant Analysis:**
        - Top merchants by spending
        - Subscription detection and tracking
        - "You've spent $345 at Amazon this quarter"
    - **Export Reports:**
        - PDF reports with charts
        - Customizable date ranges and sections
        - Shareable with accountant/partner

Technical Requirements:

- Charting library (Recharts, Chart.js, D3.js)
- Data aggregation and analysis
- PDF generation (jsPDF or server-side)
- Caching for performance

Success Metrics:

- 40%+ of users view analytics monthly
- Average time in analytics section >2 minutes
- Reports exported by 20%+ of users

2.5 Multi-Currency Support

Priority: Medium

Complexity: High

User Value: Medium (High for travelers and expats)

Description: Full support for multiple currencies within and across projects, with automatic conversion and rate tracking.

Features:

- **Multi-Currency Projects:**
  - Each project can have its own currency
  - "Personal" in USD, "Travel" in EUR
- **Multi-Currency Wallets:**
  - Wallet can have currency different from project
  - Useful for foreign currency accounts
- **Exchange Rate Handling:**
  - Fetch real-time rates (API: exchangerate-api.io or similar)
  - Manual rate entry for offline/custom rates
  - Historical rate storage for accurate reporting
- **Currency Conversion:**
  - Automatic conversion when transferring between different-currency wallets
  - Shows converted amount: "$100 (€92.50 at 0.925 rate)"
  - Exchange fee tracking
- **Consolidated Views:**
  - Dashboard can show all projects in single currency
  - "Total net worth: $15,000 (converted from multiple currencies)"
  - Currency selector in settings
- **Travel Mode:**
  - Temporary currency switch for trip
  - Tracks foreign spending
  - Converts back to home currency

Technical Requirements:

- Exchange rate API integration
- Currency conversion logic
- Historical rate storage
- Multi-currency display components

Success Metrics:

- 15%+ of users enable multi-currency
- Travelers rate feature highly valuable
- Accurate currency conversions

2.6 Receipt Scanning & OCR

Priority: Low-Medium

Complexity: High

User Value: Medium

Description: Take photo of receipt to automatically extract and log transaction details.

Features:

- **Camera Integration:**
  - In-app camera or photo library import
  - Guides user to capture clear photo
- **OCR Extraction:**
  - Reads merchant name, date, total amount, items
  - Handles various receipt formats
  - Works with printed and handwritten receipts (lower accuracy)
- **Auto-Fill Transaction:**
  - Pre-fills transaction form with extracted data
  - User reviews and corrects before saving
  - Attaches receipt image to transaction
- **Receipt Storage:**
  - Stores receipt images with transactions
  - Searchable by merchant, date, amount
  - Export receipts for tax purposes

Technical Requirements:

- OCR library (Tesseract.js or cloud API like Google Vision)
- Image processing (compression, enhancement)
- Cloud storage for receipt images (Supabase Storage)
- Privacy: User controls whether to store images

Success Metrics:

- 30%+ of users try receipt scanning
- Extraction accuracy >70% (user review step mitigates errors)
- Users find receipts easily when needed

Phase 3 Features (Months 7-12 Post-Launch)

3.1 Achievements & Gamification

Priority: Low-Medium

Complexity: Low-Medium

User Value: Medium (High for engagement)

Description: Reward users for good financial habits with badges, streaks, and milestones to increase engagement and motivation.

Features:

- **Badges:**
  - "Budget Master" - Stayed under budget for 3 months
  - "Debt Destroyer" - Paid off a debt completely
  - "Streak Keeper" - Logged transactions 30 days in a row
  - "Savings Champion" - Reached savings goal
  - "Organization Pro" - Created 3+ projects
- **Streaks:**
  - Daily transaction logging streak
  - Budget adherence streak
  - Debt payment streak
  - Visual streak counter on dashboard
- **Milestones:**
  - First 100 transactions logged
  - First month under budget
  - $1,000 saved
  - 50% debt paid off
  - Celebration animations and messages
- **Progress Levels:**
  - User levels: Beginner → Budgeter → Master → Financial Guru
  - Unlock features or insights at higher levels
  - Leaderboard (opt-in, anonymous)

Success Metrics:

- Retention increase of 10%+ vs. control group
- Users with achievements log 20%+ more transactions
- Positive sentiment in feedback

3.2 Shared Projects & Collaboration

Priority: Medium

Complexity: Very High

User Value: High (for couples and families)

Description: Invite others to view or edit shared projects like household budgets or family savings pools.

Features:

- **Project Sharing:**
  - Invite via email or link
  - Permissions: View-only, Can Edit, Admin
- **Collaborative Editing:**
  - Multiple users can add transactions
  - Real-time sync (Supabase Realtime)
  - Activity feed: "Alice added $50 grocery expense"
- **Split Expenses:**
  - Mark transactions as split
  - Assign portions to each member
  - Track who owes whom
- **Communication:**
  - Comments on transactions
  - Notes for shared context
- **Conflict Resolution:**
  - Last-write-wins or approval workflows
  - Prevent simultaneous edits of same transaction

Technical Requirements:

- Project members table (user_id, project_id, role)
- Permission system and RLS updates
- Real-time collaboration logic
- Conflict resolution
- Notification system for activity

Success Metrics:

- 20%+ of users share at least one project
- Shared projects have 30%+ more transactions (more tracking)
- High satisfaction from couples/families

3.3 Bill Reminders & Payment Tracking

Priority: Medium

Complexity: Medium

User Value: Medium-High

Description: Smart reminders for upcoming bills, payment status tracking, and late payment alerts.

Features:

- **Bill Calendar:**
  - Visual calendar showing due dates
  - Color-coded by status: Upcoming, Due Today, Overdue, Paid
- **Payment Tracking:**
  - Mark bills as paid/unpaid
  - Link transactions to bills automatically
  - "Rent due in 3 days - $1,200"
- **Smart Reminders:**
  - Notifications 3 days before, 1 day before, on due date
  - Customizable timing per bill
  - Snooze or mark paid directly from notification
- **Bill History:**
  - See past payments for each bill
  - Detect increases: "Electric bill up 15% from last month"
- **Auto-Pay Detection:**
  - Learns which bills auto-pay vs. manual
  - Different reminder styles for each

Success Metrics:

- 50%+ of users enable bill reminders
- Late payments reduce by 30%
- High engagement with bill calendar

3.4 Investment Tracking

Priority: Low

Complexity: High

User Value: Medium (High for investors)

Description: Track investment accounts, portfolios, and returns alongside cash management.

Features:

- **Investment Wallets:**
  - Special wallet type for investments
  - Track stock portfolios, mutual funds, crypto
- **Balance Tracking:**
  - Manual entry or API integration (future)
  - Shows current value, cost basis, gains/losses
- **Portfolio View:**
  - Asset allocation breakdown
  - Performance over time
  - Dividend/interest tracking
- **Net Worth Calculation:**
  - Combines cash + investments
  - Total net worth dashboard

Technical Requirements:

- Investment data model
- API integration for live prices (optional)
- Return calculation logic

Success Metrics:

- 25%+ of users with investments use this feature
- Net worth view highly rated

3.5 Tax Preparation Assistance

Priority: Low

Complexity: High

User Value: High (Seasonally)

Description: Help users prepare for tax filing with categorized expense reports, deduction tracking, and export for accountants.

Features:

- **Tax Categories:**

    - Mark categories as tax-deductible
    - Business expense tracking
    - Charitable donations
- **Year-End Reports:**
    - Annual summary of income and deductible expenses
    - Organized by tax category
    - Export in accountant-friendly format
- **Mileage Tracking:**
    - Log business/charity mileage
    - Calculate deduction value
- **Receipt Organization:**
    - Tag receipts as needed for taxes
    - Easy export of all tax-related receipts

Success Metrics:

- 40%+ of freelance/business users use tax features
- Reduces time spent on tax prep by 50%

Phase 4 Features (Year 2+)

4.1 API & Integrations

- **Open API:** Allow third-party apps to integrate with Headroom
- **Zapier/IFTTT:** Automate workflows
- **Accounting Software:** Sync with QuickBooks, Xero for business users
- **Banking APIs:** Direct bank account connections (Plaid, Yodlee)

4.2 Financial Education & Tips

- **In-App Learning:** Articles, videos on budgeting, debt reduction, saving
- **Personalized Tips:** Based on user's spending patterns
- **Goal-Based Guidance:** Help users set and achieve financial goals

4.3 Premium Subscription Features

Free Tier (Always):

- Unlimited transactions
- Up to 3 projects
- Basic budgets and debt tracking
- Offline functionality
- Core features

Premium Tier ($5-10/month):

- Unlimited projects
- Advanced analytics and reports
- Receipt scanning and storage
- Priority customer support
- Early access to new features
- CSV/PDF export (expanded)
- Multi-currency support
- Shared projects (collaboration)

4.4 Native Mobile Apps

- **iOS App:** Swift/SwiftUI for better iOS performance
- **Android App:** Kotlin for better Android performance
- **Widgets:** Home screen widgets showing Free to Spend

4.5 AI Financial Advisor (Long-Term Vision)

- **Conversational AI:** Chat with AI about finances
- **Personalized Advice:** Based on spending patterns and goals
- **Scenario Planning:** "What if I increased my debt payment to $500/month?"
- **Goal Setting:** AI helps set realistic financial goals

Feature Prioritization Criteria

When evaluating future features, consider:

1. **User Impact:** How many users benefit? How much value does it add?
2. **Alignment:** Does it support core vision (effortless financial tracking)?
3. **Complexity:** Development time and maintenance burden
4. **Differentiation:** Does it make Headroom unique vs. competitors?
5. **Monetization:** Can it support premium tier or drive retention?
6. **User Demand:** How often is it requested in feedback?

Decision Framework:

- High Impact + Low Complexity = Do Soon (Phase 2)
- High Impact + High Complexity = Strategic Investment (Phase 3)
- Low Impact + Low Complexity = Nice-to-Have (Phase 4)
- Low Impact + High Complexity = Skip

Feedback-Driven Roadmap

Post-launch, the roadmap will be heavily influenced by:

- User feedback and feature requests
- Usage analytics (which features are most used)
- Competitive analysis
- Market trends
- Technical feasibility

We commit to:

- Quarterly roadmap reviews
- Transparent communication about priorities
- User voting on feature requests (future)
- Rapid iteration based on real usage data

Long-Term Vision (3-5 Years)

Headroom becomes:

1. **The default budgeting tool** for budget-conscious individuals in emerging markets
2. **A complete financial OS** managing cash, debt, savings, and investments
3. **A collaborative platform** for families and small businesses
4. **An educational resource** teaching financial literacy through use
5. **A sustainable business** supporting continued development through premium features

6. Technical Architecture

Frontend Stack

Framework & Build Tool:

- **React 18** - Component-based UI with Hooks
- **Vite** - Fast build tool and dev server
- **TypeScript** - Type safety and better developer experience

Styling:

- **Tailwind CSS** - Utility-first CSS framework
- **PostCSS** - CSS processing
- **CSS Modules** (Optional) - For component-specific styles

Animation:

- **Framer Motion** - Declarative animations and gestures
- **react-spring** (Alternative) - Spring-physics based animations

State Management:

- **Redux Toolkit** - Global state management with built-in best practices
    - Handles: User session, active project, wallet balances, sync status
    - Redux Persist - Persist state to localStorage
- **Context API** (Alternative) - For simpler state needs
- **React Query** (Optional) - Server state management and caching

Routing:

- **React Router v6** - Client-side routing
- Nested routes for project contexts
- Protected routes for authenticated users

Local Storage:

- **Dexie.js** - IndexedDB wrapper for robust offline storage
- **localForage** (Alternative) - Unified API for localStorage/IndexedDB

Forms & Validation:

- **React Hook Form** - Performant form handling
- **Zod** or **Yup** - Schema validation

PWA & Offline:

- **Workbox** - Service Worker management
- **Web App Manifest** - PWA configuration
- **Background Sync API** - Offline transaction queuing

Utilities:

- **date-fns** - Date manipulation
- **currency.js** - Precise currency calculations
- **clsx** or **classnames** - Conditional className management
- **react-hot-toast** - Toast notifications
- **lucide-react** - Icon library

Backend Stack

Runtime & Framework:

- **Node.js** (v18+) - JavaScript runtime
- **Express.js** - Web application framework
- **TypeScript** - Type-safe backend

Database:

- **PostgreSQL** (v14+) - Primary relational database
- Hosted via **Supabase** - Managed PostgreSQL with additional features

ORM:

- **Prisma** - Type-safe database ORM
    - Schema definition and migrations
    - Auto-generated TypeScript types
    - Query builder with excellent DX

Authentication:

- **Supabase Auth** - Authentication service
    - OAuth providers: Google, Apple, Microsoft
    - Email/password authentication
    - JWT token management
    - Session handling
    - Row Level Security (RLS) integration

Real-time & Sync:

- **Supabase Realtime** - WebSocket-based real-time updates

    - Automatic data synchronization across devices
    - Built on PostgreSQL's replication features
    - Subscribe to database changes

API Design:

- **RESTful API** - Standard HTTP methods
- **JSON** - Data exchange format
- **Express middleware** - Request processing pipeline

Validation & Security:

- **Zod** - Runtime type checking and validation
- **express-validator** (Alternative) - Request validation
- **helmet** - Security headers
- **cors** - CORS configuration
- **express-rate-limit** - API rate limiting

Background Jobs (Future):

- **node-cron** or **Bull** - Scheduled tasks (recurring transaction generation, notifications)

Infrastructure

Hosting & Deployment

Frontend Hosting:

- **Vercel**
    - Automatic deployments from Git
    - Edge network (CDN)
    - Serverless functions support
    - Built-in analytics
    - Free tier suitable for MVP
- **Netlify** (Alternative) - Similar features

Backend Hosting:

- **Render**
    - Easy PostgreSQL setup
    - Automatic HTTPS
    - Auto-deploy from Git
    - Free tier available
    - Integrated with Supabase

- **Railway** (Alternative) - Developer-friendly platform

Database Hosting:

- **Supabase**
  - Managed PostgreSQL
  - Built-in Auth, Realtime, Storage
  - Generous free tier (500MB database, 2GB bandwidth)
  - Automatic backups
  - Dashboard for database management
  - Row Level Security (RLS) policies

CDN:

- **Vercel Edge Network** - Included with Vercel hosting

Monitoring & Analytics:

- **Vercel Analytics** - Frontend performance
- **Sentry** - Error tracking (frontend + backend)
- **Supabase Dashboard** - Database metrics and logs
- **Google Analytics** (Optional) - User behavior tracking

Data Models

Database Schema (PostgreSQL via Prisma)

Relationships Overview:

User (1) → (Many) Project

Project (1) → (Many) Wallet

Project (1) → (Many) Category

Project (1) → (Many) Budget

Project (1) → (Many) Debt

Project (1) → (Many) Transaction

Wallet (1) → (Many) Transaction (as source or destination)

Category (1) → (Many) Transaction

Debt (1) → (Many) Transaction (as repayment links)

User

Purpose: Represents an authenticated user account

```
model User {
  id               String      @id @default(uuid())
  email            String      @unique
  name             String?
  auth_provider    String      // 'google', 'apple', 'microsoft', 'email'
  auth_provider_id String?     // Provider's user ID
  created_at       DateTime    @default(now())
  updated_at       DateTime    @updatedAt
  last_login       DateTime?

  // Settings (stored as JSON for flexibility)
  settings         Json?       // { currency, theme, notifications, etc. }

  // Relationships
  projects         Project[]

  @@index([email])
}
```

Settings JSON Structure:

```
{
  currency: string,              // 'USD', 'UGX', 'EUR', etc.
  theme: 'light' | 'dark',       // Future: dark mode
  language: string,              // 'en', 'sw', etc.
  notifications: {
    enabled: boolean,
    daily_reminder: boolean,
    daily_reminder_time: string, // '07:00'
    weekly_summary: boolean,
    weekly_summary_day: number,  // 0-6 (Sunday-Saturday)
    weekly_summary_time: string,
    budget_alerts: boolean,
    quiet_hours: {
      enabled: boolean,
      start: string,             // '22:00'
      end: string                // '07:00'
    }
  }
}
```

Project

Purpose: Isolated financial context (e.g., Personal, Business, Family)

```
model Project {
  id              String      @id @default(uuid())
  user_id         String
  name            String
  is_primary      Boolean     @default(false)
  tracking_period  String     @default('monthly') // 'daily', 'weekly', 'monthly'
  currency        String      @default('USD')
  created_at      DateTime    @default(now())
  updated_at      DateTime    @updatedAt
  last_accessed   DateTime    @default(now())

  // Relationships
  user            User        @relation(fields: [user_id], references: [id], onDelete: Cascade)
  wallets         Wallet[]
  categories      Category[]
  transactions    Transaction[]
  budgets         Budget[]
  debts           Debt[]

  @@index([user_id])
  @@index([user_id, is_primary])
}
```

Notes:

- is_primary: Only one project per user should be primary (enforced in application logic)
- last_accessed: Updated when user switches to this project
- currency: Can differ from user's default currency (for multi-currency support in future)

Wallet

Purpose: Money storage within a project (e.g., Cash, Bank Account)

```
model Wallet {
  id              String      @id @default(uuid())
  project_id      String
  name            String
  type            String?     // 'cash', 'bank', 'mobile_money', 'credit_card', 'custom'
  icon            String?     // Icon identifier or emoji
  color           String?     // Hex color code
  initial_balance  Decimal     @default(0) @db.Decimal(15, 2)
  current_balance  Decimal     @default(0) @db.Decimal(15, 2)
  is_default      Boolean     @default(false)
  is_archived     Boolean     @default(false)
  created_at      DateTime    @default(now())
  updated_at      DateTime    @updatedAt

  // Relationships
  project          Project     @relation(fields: [project_id], references: [id], onDelete: Cascade)
```

```
  transactions_from     Transaction[] @relation("SourceWallet")
  transactions_to       Transaction[] @relation("DestinationWallet")

  @@index([project_id])
  @@index([project_id, is_default])
  @@index([project_id, is_archived])
}
```

Notes:

- current_balance: Calculated field, updated on transaction CRUD
- is_default: Only one wallet per project should be default (enforced in application logic)
- is_archived: Hidden from UI but preserves history

Category

Purpose: Spending/income classification within a project

```
model Category {
  id              String      @id @default(uuid())
  project_id      String
  name            String
  type            String      @default('expense') // 'income', 'expense'
  icon            String?     // Icon identifier or emoji
  color           String?     // Hex color code
  is_standard     Boolean     @default(false) // True for predefined categories
  is_hidden       Boolean     @default(false)
```

```
  display_order    Int?        // For custom ordering

  created_at       DateTime    @default(now())

  updated_at       DateTime    @updatedAt


  // Relationships

  project          Project      @relation(fields: [project_id], references: [id], onDelete:
Cascade)

  transactions     Transaction[]

  budgets          Budget[]


  @@index([project_id])

  @@index([project_id, type])

  @@index([project_id, is_hidden])

}
```

Standard Categories (Seeded on Project Creation):

- Expense: Food, Transport, Shopping, Bills, Entertainment, Health, Other
- Income: Salary, Gift, Refund, Investment, Other

Notes:

- is_standard: Cannot be deleted, only hidden
- is_hidden: Removed from UI but transactions remain linked

Transaction

Purpose: Individual financial record (income, expense, transfer)

```
model Transaction {
  id                  String      @id @default(uuid())
  project_id          String
  type                String      // 'income', 'expense', 'transfer', 'adjustment'
  amount              Decimal     @db.Decimal(15, 2)
  description         String?
  date                DateTime    @default(now())

  // Wallet relationships
  source_wallet_id      String?     // For expenses and transfers (from)
  destination_wallet_id String?     // For income and transfers (to)

  // Category (not applicable for transfers)
  category_id         String?

  // Recurring transaction info
  is_recurring          Boolean     @default(false)
  recurring_pattern_id  String?     // Groups recurring transactions
  recurring_frequency   String?     // 'daily', 'weekly', 'monthly', 'yearly'
  recurring_start_date  DateTime?
  recurring_end_date    DateTime?
  recurring_needs_confirm Boolean   @default(false)
```

```
  // Transfer-specific
  transfer_fee          Decimal?    @db.Decimal(15, 2)

  // Debt linkage
  debt_id               String?

  // Sync metadata
  is_synced             Boolean     @default(false)
  local_timestamp       DateTime    @default(now())
  synced_at             DateTime?

  created_at            DateTime    @default(now())
  updated_at            DateTime    @updatedAt

  // Relationships
  project               Project     @relation(fields: [project_id], references: [id], onDelete:
Cascade)
  source_wallet         Wallet?     @relation("SourceWallet", fields: [source_wallet_id],
references: [id], onDelete: SetNull)
  destination_wallet    Wallet?     @relation("DestinationWallet", fields:
[destination_wallet_id], references: [id], onDelete: SetNull)
  category              Category?   @relation(fields: [category_id], references: [id],
onDelete: SetNull)
  debt                  Debt?       @relation(fields: [debt_id], references: [id], onDelete:
SetNull)

  @@index([project_id])
  @@index([project_id, date])
```

```
  @@index([project_id, type])
  @@index([project_id, category_id])
  @@index([source_wallet_id])
  @@index([destination_wallet_id])
  @@index([recurring_pattern_id])
  @@index([is_synced])
}
```

Transaction Types:

- income: Money received (destination_wallet_id required, category_id required)
- expense: Money spent (source_wallet_id required, category_id required)
- transfer: Move between wallets (both wallet IDs required, category_id null)
- adjustment: Manual balance correction (one wallet ID, special category)

Recurring Logic:

- recurring_pattern_id: UUID shared by all transactions from same pattern
- First transaction in pattern stores the configuration
- Generated transactions reference original via recurring_pattern_id

Budget

Purpose: Spending limits for categories or overall project

```
model Budget {
  id              String      @id @default(uuid())
  project_id      String
  category_id     String?     // Null = overall project budget
  limit_amount    Decimal     @db.Decimal(15, 2)
  period          String      @default('monthly') // 'daily', 'weekly', 'monthly'
```

```
  // Alert thresholds
  alert_at_80      Boolean     @default(true)
  alert_at_100     Boolean     @default(true)
  alert_at_120     Boolean     @default(false)

  created_at       DateTime    @default(now())
  updated_at       DateTime    @updatedAt

  // Relationships
  project          Project     @relation(fields: [project_id], references: [id], onDelete:
Cascade)
  category         Category?   @relation(fields: [category_id], references: [id], onDelete:
Cascade)

  @@unique([project_id, category_id, period])
  @@index([project_id])
}
```

Notes:

- One budget per category per period per project
- Overall budget has category_id = null

Debt

Purpose: Track loans and repayment progress

```
model Debt {
  id               String        @id @default(uuid())
  project_id       String
  name             String
  total_amount     Decimal       @db.Decimal(15, 2)
  remaining_amount Decimal       @db.Decimal(15, 2)
  interest_rate    Decimal?      @db.Decimal(5, 2) // Annual percentage
  minimum_payment  Decimal?      @db.Decimal(15, 2)
  due_date         DateTime?     // Next payment due date

  created_at       DateTime      @default(now())
  updated_at       DateTime      @updatedAt

  // Relationships
  project          Project       @relation(fields: [project_id], references: [id], onDelete:
Cascade)
  repayments       Transaction[] // Transactions linked to this debt

  @@index([project_id])
}
```

Notes:

- remaining_amount: Updated when transactions are linked as repayments
- Repayment transactions have debt_id set and reduce remaining_amount

Data Flow Architecture

Offline-First Pattern

Write Flow (User Creates Transaction):

1. **Local Write (Instant)**

   User Action → React Component

   → Redux Action

   → Dexie.js (IndexedDB)

   → UI Update (Optimistic)

2. **Background Sync (When Online)**

   Service Worker → Detect Connection

   → Read Pending Queue (IndexedDB)

   → POST /api/transactions

   → Update is_synced flag

   → Clear from queue

3. **Conflict Resolution**

   a. Last-write-wins based on local_timestamp
   b. Server timestamp used for tie-breaking
   c. Deleted items marked with deleted_at instead of hard delete

Read Flow (User Views Data):

1. **Initial Load**

   Component Mount → Check IndexedDB

   → Render Cached Data

   → Fetch from API (background)

   → Merge and Update UI

2. **Real-time Updates (Supabase Realtime)**

   Database Change → Supabase Realtime → WebSocket

   → Redux Store Update

   → UI Re-render

State Management Architecture

Redux Store Structure:

```
{
  auth: {
    user: User | null,
    token: string | null,
    isAuthenticated: boolean
  },
  projects: {
    list: Project[],
    active: Project | null,
    loading: boolean,
    error: string | null
  },
  wallets: {
    byProject: {
      [projectId]: Wallet[]
    },
    loading: boolean
  },
```

```
categories: {
  byProject: {
    [projectId]: Category[]
  },
  loading: boolean
},
transactions: {
  byProject: {
    [projectId]: Transaction[]
  },
  pendingSync: string[], // Transaction IDs
  loading: boolean
},
budgets: {
  byProject: {
    [projectId]: Budget[]
  }
},
debts: {
  byProject: {
    [projectId]: Debt[]
  }
},
sync: {
  status: 'idle' | 'syncing' | 'error',
  lastSyncTime: Date | null,
```

```
    pendingCount: number
  },
  ui: {
    activeScreen: string,
    modals: {
      addTransaction: boolean,
      projectSwitcher: boolean
    },
    toast: {
      message: string,
      type: 'success' | 'error' | 'info'
    }
  }
}
```

API Architecture

RESTful Endpoints:

Authentication:

POST /api/auth/signup - Create account

POST /api/auth/signin - Email/password login

POST /api/auth/oauth/google - OAuth (handled by Supabase)

POST /api/auth/oauth/apple

POST /api/auth/signout - Logout

GET /api/auth/me - Get current user

Projects:

GET /api/projects - List user's projects

POST /api/projects - Create project

GET /api/projects/:id - Get project details

PATCH /api/projects/:id - Update project

DELETE /api/projects/:id - Delete project

POST /api/projects/:id/switch - Switch active project

Wallets:

GET /api/projects/:projectId/wallets - List wallets

POST /api/projects/:projectId/wallets - Create wallet

GET /api/wallets/:id - Get wallet

PATCH /api/wallets/:id - Update wallet

DELETE /api/wallets/:id - Delete wallet

POST /api/wallets/:id/adjust-balance - Manual adjustment

Categories:

GET /api/projects/:projectId/categories - List categories

POST /api/projects/:projectId/categories - Create category

PATCH /api/categories/:id - Update category

DELETE /api/categories/:id - Delete category (custom only)

Transactions:

GET /api/projects/:projectId/transactions - List transactions (with filters)

POST /api/projects/:projectId/transactions - Create transaction

POST /api/projects/:projectId/transactions/bulk - Batch create (sync)

GET /api/transactions/:id - Get transaction

PATCH /api/transactions/:id - Update transaction

DELETE /api/transactions/:id - Delete transaction

Budgets:

GET /api/projects/:projectId/budgets - List budgets

POST /api/projects/:projectId/budgets - Create budget

PATCH /api/budgets/:id - Update budget

DELETE /api/budgets/:id - Delete budget

Debts:

GET /api/projects/:projectId/debts - List debts

POST /api/projects/:projectId/debts - Create debt

PATCH /api/debts/:id - Update debt

DELETE /api/debts/:id - Delete debt

Sync:

POST /api/sync/transactions - Sync pending transactions

GET /api/sync/status - Get sync status

Request/Response Format:

// Request

{

```
  data: {
    // Request body
  },
  meta: {
    timestamp: number,
    clientId: string
  }
}

// Success Response
{
  data: {
    // Response data
  },
  meta: {
    timestamp: number
  }
}

// Error Response
{
  error: {
    code: string,
    message: string,
    details?: any
  },
```

```
  meta: {
    timestamp: number
  }
}
```

Security Architecture

Authentication & Authorization

JWT Tokens (Supabase Auth):

- Access token: Short-lived (1 hour), sent with each request
- Refresh token: Long-lived (7 days), used to get new access token
- Stored in HTTP-only cookies (backend) or secure localStorage (frontend)

Row Level Security (RLS) Policies:

```
-- Users can only see their own data
CREATE POLICY user_isolation ON users
  FOR ALL
  USING (auth.uid() = id);

-- Users can only access their own projects
CREATE POLICY project_access ON projects
  FOR ALL
  USING (auth.uid() = user_id);

-- Cascade: Wallets, Categories, Transactions, Budgets, Debts
-- All inherit project ownership check
CREATE POLICY wallet_access ON wallets
```

```
  FOR ALL
  USING (
    EXISTS (
      SELECT 1 FROM projects
      WHERE projects.id = wallets.project_id
      AND projects.user_id = auth.uid()
    )
  );
```

API Security:

- All endpoints require valid JWT (except auth endpoints)
- User ID extracted from JWT, not request body
- Project ownership verified on all project-scoped operations
- Rate limiting: 100 requests/minute per user
- CORS configured for frontend domain only

Data Privacy

Encryption:

- HTTPS/TLS for all communication
- Database connections encrypted
- Supabase provides encryption at rest
- Passwords hashed with bcrypt (handled by Supabase Auth)

Data Isolation:

- Users cannot access other users' data (enforced by RLS)
- Projects are completely isolated from each other
- No shared data between users in MVP

Performance Optimization

Frontend Performance

Code Splitting:

- Route-based code splitting (React.lazy)
- Component-level splitting for heavy components
- Vendor bundle optimization

Caching Strategy:

- Service Worker caches static assets
- API responses cached in IndexedDB
- Redux Persist for state persistence
- Cache invalidation on data changes

Rendering Optimization:

- React.memo for expensive components
- useMemo/useCallback for derived data
- Virtual scrolling for long lists (react-window)
- Lazy loading for images

Bundle Size:

- Tree shaking unused code
- Minification and compression
- Target: <200KB initial bundle

Backend Performance

Database Optimization:

- Indexes on frequently queried fields
- Pagination for list endpoints (limit/offset)
- Query optimization (avoid N+1)
- Connection pooling

Caching:

- In-memory cache for static data (categories, budgets)
- Redis (future) for session storage and caching

API Optimization:

- Batch endpoints for sync operations
- Compression (gzip)
- CDN for static assets

Backup & Recovery

Database Backups:

- Supabase automatic daily backups (retained for 7 days on free tier)
- Point-in-time recovery available on paid tiers
- Manual backups before major migrations

Data Export:

- Users can export their data as CSV
- Includes all transactions, budgets, debts
- Future: Full data export in JSON format

Disaster Recovery:

- Database replica (Supabase provides)
- Application redeployment plan
- Data restoration procedures documented

Development Workflow

Version Control:

- Git with GitHub
- Branch strategy: main (production), develop (staging), feature branches
- Pull request reviews required
- Conventional commits

CI/CD Pipeline:

- GitHub Actions or Vercel/Render built-in CI/CD
- Automated testing on PR
- Automatic deployment to staging on develop branch
- Manual approval for production deployment

Testing Strategy:

- Unit tests: Jest + React Testing Library
- Integration tests: API endpoint testing
- E2E tests: Playwright or Cypress (limited for MVP)
- Coverage target: >70% for core logic

Code Quality:

- ESLint + Prettier
- TypeScript strict mode
- Pre-commit hooks (Husky)
- Code reviews

Scalability Considerations

Current Architecture (MVP):

- Supports: 1,000-10,000 users
- Database: PostgreSQL on Supabase free tier (500MB)
- Backend: Single server instance

Future Scaling:

- Horizontal scaling: Multiple backend instances
- Database: Upgrade Supabase tier or move to managed PostgreSQL
- Caching: Redis for session and query caching
- Queue system: Bull/BullMQ for background jobs

Third-Party Services

Required (MVP):

- Supabase: Database, Auth, Realtime
- Vercel: Frontend hosting
- Render: Backend hosting

Optional (Future):

- Sentry: Error tracking
- Firebase Cloud Messaging: Push notifications
- Stripe: Payment processing (for premium features)

- Twilio SendGrid: Transactional emails

API Rate Limiting Strategy

Rate Limits:

- Authentication endpoints: 5 requests/minute
- Read endpoints: 100 requests/minute
- Write endpoints: 50 requests/minute
- Sync endpoint: 10 requests/minute

Implementation:

- Express rate limit middleware
- User-based (from JWT)
- Redis store for distributed rate limiting (future)

Response Headers:

X-RateLimit-Limit: 100

X-RateLimit-Remaining: 95

X-RateLimit-Reset: 1640000000

Database Migration Strategy

Prisma Migrations:

 # Create migration 'add_projects_table'

npx prisma migrate dev --name add_projects_table

 # Apply to production

npx prisma migrate deploy

Migration Best Practices:

- Always backup before migration
- Test migrations on staging first
- Reversible migrations when possible
- Seed default categories on new projects. Default categories should have a name, type, icon, and have is_standard set to true.

Environment Variables

Frontend (.env):

VITE_API_URL=https://api.headroom.com

VITE_SUPABASE_URL=https://xxxxx.supabase.co

VITE_SUPABASE_ANON_KEY=xxxxx

VITE_ENABLE_ANALYTICS=true

Backend (.env):

NODE_ENV=production

PORT=3000

DATABASE_URL=postgresql://user:pass@host:5432/db

SUPABASE_URL=https://xxxxx.supabase.co

SUPABASE_SERVICE_ROLE_KEY=xxxxx

JWT_SECRET=xxxxx

CORS_ORIGIN=https://headroom.com

SENTRY_DSN=xxxxx

7. User Experience & Design

Design Principles

1. **Mobile-First with Context Awareness**
    - All interfaces designed for touch and small screens first
    - Project context always visible (header shows current project)
    - Wallet information present but not intrusive (cards, not constant selection)
    - Category-first navigation replaces time-based as primary pattern

2. **Minimal Input with Smart Defaults**
    - Reduce typing through intelligent defaults and memory
    - Default project remembered across sessions
    - Default wallet per project speeds up transaction entry
    - Category pre-selection when navigating from category view
    - Previous values suggested where appropriate

3. **Immediate Feedback**
    - Visual confirmation for all actions
    - Real-time wallet balance updates
    - Animated Free to Spend recalculation
    - Sync status always visible
    - Loading states that inform, not just block

4. **Forgiving & Flexible**
    - Easy undo and edit capabilities at every level
    - Manual wallet balance adjustments when needed
    - Can pause recurring transactions without deleting
    - Project switching without data loss
    - Offline-first means never losing work

5. **Calm Technology**
    - Helpful without being intrusive
    - Contextual tutorials only when needed
    - Notifications intelligently timed (morning/evening)
    - Maximum 2 nudges per day
    - Quiet hours respect user's schedule
    - Progressive disclosure hides complexity until needed

6. **Hierarchical Clarity**
    - Clear visual hierarchy: Project → Wallets/Categories → Transactions
    - Free to Spend is always the hero element
    - Information density appropriate to screen importance
    - Consistent patterns across project contexts

Key Screens

1. Dashboard (Home Screen)

Purpose: At-a-glance financial overview for the active project

Layout Structure:

Header Section:

- Left: Current project name (tappable dropdown)
    - Shows project name in medium-bold text
    - Dropdown arrow indicates interactivity
    - Tap opens project switcher
- Right: Sync status icon + Settings icon
    - Sync: Animated when syncing, checkmark when complete, warning if failed
    - Settings: Leads to app settings

Hero Section (Free to Spend):

- Largest element on screen (takes ~30% of viewport)
- Amount displayed in extra-large, bold typography
- Currency symbol smaller but visible
- Period indicator below: "This Month" or "This Week" or "Today"
- Color-coded background card:
    - Green gradient when healthy (>20% of budget remaining)
    - Yellow gradient when approaching limit (5-20% remaining)
    - Red gradient when over budget (<0%)
- Subtle pulsing animation on load to draw attention
- Tap for detailed breakdown modal

Wallet Cards Section:

- Horizontal scrollable row (if >3 wallets) or grid (if ≤3)
- Each wallet card shows:
    - Wallet icon (customizable per wallet type)
    - Wallet name (e.g., "Cash", "Bank Account")
    - Current balance (medium-large, bold)
    - Small trend indicator: "↑ $50 this week" or "↓ $120 this week"
- Cards have subtle shadow, rounded corners
- Tap wallet card: Quick actions (View Details, Transfer, Edit, Transactions)
- "+" button at end to add new wallet

Quick Stats Section:

- 2-3 info cards in grid:

    - **Spent This Period:** Amount + percentage of budget
    - **Upcoming Bills:** Count + total amount due soon
    - **Top Spending Category:** Category name + amount
- Minimal design: icon + label + value
- Tap any card for detailed view

Recent Activity (Optional, can be hidden):

- Last 3 transactions
- Compact list: icon, category, amount, time
- "View All" link → Opens Reports
- Can be disabled in settings for cleaner dashboard

Navigation Hints:

- Subtle text at bottom: "Swipe left for Categories" (dismissible after first use)
- Or category preview cards that encourage tap/swipe

Spacing & Whitespace:

- Generous padding between sections (16-24px)
- Card-based layout with consistent 12px border radius
- Background: Off-white (#F8F9FA) to reduce eye strain
- Cards: Pure white (#FFFFFF) with subtle shadows (0 2px 8px rgba(0,0,0,0.08))

2. Categories View (Primary Navigation)

Purpose: Browse and manage spending by category for active project

Layout Structure:

Header:

- Shows "[Project Name] - Categories"
- Back button (if navigated from elsewhere) or hamburger menu
- Filter/Sort icon (top right): Sort by spending, alphabetical, custom order

Category Grid:

- 2-column grid on mobile (1 column on very small screens <360px)
- 3-4 columns on tablet/desktop
- Each category card:
    - **Top:** Category icon (large, colorful)
    - **Middle:** Category name (bold, readable size)

        - **Budget bar:** Visual progress indicator
            - Shows: "$245.80 / $500" or "No budget set"
            - Progress bar color: green (safe), yellow (warning), red (exceeded)
            - Percentage displayed: "49%" or "113% ⚠"
        - **Bottom:** Last transaction time: "2 hours ago" or "No transactions"
- Cards have hover/press states (slight scale up, shadow increase)
- Empty categories shown with muted colors/dashed border

Add Category Card:

- "+" icon prominently displayed
- "Add Category" text
- Same size as category cards
- Positioned at end of grid

Floating Action Button (FAB):

- Bottom right corner
- Primary action: "Add Transaction"
- Tap opens transaction flow (will ask for category during flow)

Empty State (No Categories):

- Illustration + text: "No categories yet"
- "Add your first category" button
- Helper text: "Organize your spending into categories"

3. Category Detail View

Purpose: Deep dive into single category spending and budget management

Layout Structure:

Header:

- Back button + Category name
- Settings icon (top right) → Category settings menu

Top Summary Card (Hero):

- Large card with category color accent
- **Main stat:** "Spent this month: $245.80"
- **Budget info:**
    - If set: Progress bar + "$245.80 / $500" + "Free to spend: $254.20"

    - If not set: "Set a budget" button
- **Quick stats row:**
    - Average per transaction: $15.36
    - Transaction count: 16 transactions
    - Biggest expense: $45.00 (with detail)

Transaction List:

- Grouped by date sections:
    - "Today" (if any)
    - "Yesterday" (if any)
    - "This Week" (if beyond yesterday)
    - "Earlier This Month"
    - Older months collapsed by default
- Each transaction:
    - Left: Category icon (smaller)
    - Center: Description/note (if any) or "No description" (muted)
    - Right: Amount (bold, color-coded)
    - Below: Wallet used + time (small, muted text)
- Swipe actions:
    - Swipe left: Edit | Delete (destructive red)
- Tap transaction: Opens transaction details modal

Empty State (No Transactions):

- Illustration + text: "No transactions in [Category] yet"
- "Add your first [Category] transaction" button

Floating Action Button:

- "Add [Category] Transaction"
- Pre-fills category in transaction flow

4. Transaction Entry Modal (Sequential Flow)

Purpose: Log income, expense, or transfer with minimal friction

Design Pattern: Sequential, Conversational

Modal Behavior:

- Slides up from bottom (mobile)
- Centered modal (desktop/tablet)
- Semi-transparent backdrop (dims background)

- Swipe down to dismiss (mobile) or "X" button

Step Display:

- Only current step fully visible and interactive (100% opacity)
- Previous steps fade to 30% opacity, scale down slightly, move up
- Progress indicator at top: "Step 2 of 6" or dots
- Can tap previous steps to go back and edit

Step Design (Each Step):

- Large header text: Question or instruction
- Main content: Input field, selection cards, or toggle
- Navigation:
    - "Next" button (bottom, full width on mobile, fixed width on desktop)
    - "Back" text button (top left, subtle)
    - "Skip" text button (top right, if step is optional)

Input Types:

- **Amount:** Large numeric keypad, currency symbol visible
- **Selection (wallets, categories):** Large touch-friendly cards in grid
- **Toggle (recurring):** iOS-style toggle with label
- **Text (notes):** Single-line or multi-line input with character count if limited
- **Date:** Native date picker overlay

Review Step:

- Card-based summary with all details
- Each detail editable (tap to go back to that step)
- Large "Save Transaction" or "Complete Transfer" button
- Cancel button (secondary style)

Completion:

- Modal slides down with ease-out animation
- Returns to previous screen
- Success toast appears briefly (3 seconds)
- Updated data visible immediately (wallet balance, Free to Spend)

5. Reports View

Purpose: Time-based transaction history and analytics for active project

Layout Structure:

Header:

- "[Project Name] - Reports"
- Period selector (prominent): Dropdown showing "This Month"
    - Options: Today, This Week, This Month, This Quarter, This Year, Custom Range
- Export button (icon, top right)

Summary Cards (Top Section):

- Grid of 4 cards (2x2 on mobile, 4x1 on desktop)
- Each card shows:
    - Icon + Label (small, muted)
    - Value (large, bold, color-coded)
    - Trend indicator (optional): "↑ 12% from last month"
- Cards: Total Income, Total Expenses, Net, Savings Rate

Insights Section (Collapsible):

- Small header: "Insights" with expand/collapse icon
- When expanded, shows 3-4 insight cards:
    - "Biggest expense: $450 - Rent (Bills)"
    - "Most frequent: Food (23 transactions)"
    - "Busiest day: Friday (8 transactions)"
    - "Most used wallet: Mobile Money (45%)"
- Can swipe horizontally on mobile

Filter Bar:

- Horizontal scrollable row of filter chips
- Active filters shown with "X" to remove
- "All Filters" button opens full filter modal:
    - Search by description
    - Date range picker (calendar)
    - Category multi-select (checkboxes)
    - Wallet multi-select (checkboxes)
    - Amount range (min/max sliders)
    - Type: All / Income / Expense / Transfer (radio buttons)
- "Apply Filters" button
- "Clear All" button if filters active

Transaction List:

- Chronological, grouped by date
- Date headers: "Today", "Yesterday", "Dec 15", etc.

- Each transaction row:
    - Left: Type icon (income/expense/transfer) + Category icon
    - Center:
        - Top: Description or "[Category Name]" (bold)
        - Bottom: Wallet name (small, muted)
    - Right:
        - Top: Amount (color-coded: green income, red expense, blue transfer)
        - Bottom: Time (small, muted)
- Swipe actions: Edit | Delete
- Tap: Transaction details modal

Empty State:

- "No transactions for selected filters"
- "Clear filters" button or "Add transaction" button

Loading State:

- Skeleton cards while loading
- Shimmer animation on placeholders

6. Project Switcher (Dropdown)

Purpose: Quickly switch between projects or manage them

Dropdown Design:

- Appears below header when tapped
- Overlay dims background slightly
- Rounded corners, shadow for depth

Dropdown Content:

- **Current Project (Top):**
    - Highlighted with accent color background
    - Checkmark icon on right
    - Shows: Project name (bold) + total balance + wallet count
    - "Currently active" label (small, muted)
- **Other Projects:**
    - Listed below current
    - Each shows: Project name + total balance + wallet count + last used
    - Tap any project → Immediate switch

- **Divider Line**

- **Action Buttons:**

    - "+ Create New Project" (with icon)
    - "Manage Projects" (with settings icon) → Full management screen

Interaction:

- Smooth transition animation when switching projects
- Brief loading indicator during switch (<1 second)
- New project data loads immediately
- Dropdown closes automatically after selection

7. Wallet Management Screens

Wallet List (in Settings or Dashboard):

- List of all wallets for active project
- Each wallet row:
    - Icon + Name + Balance
    - Quick actions icons: Transfer | Edit | View
- Sorted by: Default first, then by balance or custom order
- "+ Add Wallet" button at bottom

Wallet Detail View:

- Header: Wallet name + balance (large)
- Quick actions: Transfer | Adjust Balance | Settings
- Balance history chart (optional for MVP)
- Transaction list filtered to this wallet
- Filters: Date range, Type, Category

Add/Edit Wallet Modal:

- Wallet name input
- Icon picker (grid of icons) or emoji selector
- Color picker (color palette)
- Starting balance input (optional)
- "Set as default" toggle
- Save button

Transfer Modal:

- Follows transaction entry sequential pattern
- Steps: Amount → From Wallet → To Wallet → Fee (optional) → Review → Confirm

8. Settings & Account

Settings Structure (Grouped Sections):

Account:

- Profile picture + Name + Email
- "Edit Profile" button
- "Sign Out" button

Project Management:

- "Manage Projects" → Full project list
- Shows current project with star icon

Preferences:

- Currency (display current, tap to change)
- Tracking period (Daily/Weekly/Monthly)
- Default wallet (per project)
- Language (future)

Notifications:

- Toggle switches for each notification type:
    - Daily spending reminders (with time picker)
    - Weekly summaries (with day/time picker)
    - Budget alerts
    - Recurring transaction confirmations
- Quiet hours toggle + time range picker

Data & Privacy:

- Export data (CSV)
- Backup status
- Last sync timestamp
- "Sync Now" button
- Clear cache
- Delete account (destructive, requires confirmation)

Tutorial & Help:

- "Reset Tutorials" button
- "Show tutorial tips" toggle
- "Help & Support" link
- "Send Feedback" link

About:

- App version
- Terms of Service
- Privacy Policy
- Open source licenses

Navigation System

Mobile (Primary Experience)

Primary Navigation Pattern: Category-First

Bottom Floating Action Button (FAB):

- Position: Bottom right (16px from edge)
- Size: 56x56px (standard FAB size)
- Color: Primary blue with white icon
- Shadow: Elevated (0 4px 12px rgba(0,0,0,0.15))

Default Behavior:

- Single tap: Opens "Add Transaction" flow
- Most common action, always accessible

Long Press (Hold ~500ms):

- Radial menu expands with 3 options arranged in arc:
    - **Dashboard** (center/top position): Home icon
    - **Categories** (left position): Grid icon
    - **Reports** (right position): Chart icon
- Each option: Icon + label
- Tap any option to navigate
- Tap outside or release to dismiss

Swipe Gestures:

- Available on main screens (Dashboard, Categories, Reports)
- Swipe implementation:

    - **From Dashboard:**
        - Swipe left → Categories view
        - Swipe right → Reports view
    - **From Categories:**
        - Swipe right → Dashboard
    - **From Reports:**
        - Swipe left → Dashboard
- Visual feedback: Screen follows finger during swipe, then animates to completion
- Threshold: 30% of screen width to trigger navigation
- Can cancel by swiping back before releasing

Header Navigation:

- Project switcher (tap project name)
- Settings icon (always visible)
- Back button (when in nested views like category detail, wallet detail)

Desktop/Tablet (Responsive Adaptation)

Persistent Sidebar Navigation:

- Left side of screen
- Width: 240px (can collapse to 60px icon-only)
- Sections:
    - **Top:** Project switcher (current project name + dropdown)
    - **Main Navigation:** Dashboard, Categories, Reports, Settings (vertical list)
    - **Bottom:** Sync status, User profile

Main Content Area:

- Right side, takes remaining width
- Responsive grid layouts (3-4 columns for categories, 2 columns for cards)
- Maximum content width: 1200px (centered if wider)

FAB Replacement:

- Prominent "Add Transaction" button in header (top right)
- Always visible, sticky on scroll
- Opens modal (centered) instead of bottom sheet

Keyboard Shortcuts (Desktop):

- Ctrl/Cmd + N: New transaction
- Ctrl/Cmd + 1/2/3: Navigate to Dashboard/Categories/Reports

- Ctrl/Cmd + K: Search/Command palette
- Escape: Close modals

Color Palette

Primary Colors

Primary (Trust & Stability):

- Main: Soft Blue (#4A90E2)
- Light: #5B9BD5
- Dark: #357ABD
- Usage: FAB, primary buttons, links, active states, project headers

Accent (Friendly Approachability):

- Main: Warm Teal (#26A69A)
- Light: #4DB6AC
- Dark: #00897B
- Usage: Secondary buttons, highlights, wallet icons, special features

Semantic Colors

Success (Positive Actions):

- Main: Gentle Green (#66BB6A)
- Light: #81C784
- Dark: #4CAF50
- Usage: Under budget indicators, income transactions, confirmation messages, positive trends

Warning (Gentle Attention):

- Main: Soft Amber (#FFA726)
- Light: #FFB74D
- Dark: #FB8C00
- Usage: Approaching budget limit, pending sync status, caution messages, 80% budget alerts

Danger (Needs Attention):

- Main: Muted Coral (#EF5350)
- Light: #E57373
- Dark: #F44336

- Usage: Over budget, expense transactions, delete actions, critical alerts, error states

Neutral Colors

Text:

- Primary: Charcoal (#2C3E50) - Main content, headings
- Secondary: Slate Gray (#546E7A) - Supporting text, labels
- Tertiary: Light Gray (#90A4AE) - Placeholder text, disabled states
- Inverted: White (#FFFFFF) - Text on dark backgrounds

Backgrounds:

- App Background: Off-white (#F8F9FA) - Reduces eye strain
- Card Background: Pure White (#FFFFFF) - Content containers
- Hover/Active: Light Blue Gray (#ECEFF1) - Interactive states
- Disabled: Very Light Gray (#E0E0E0) - Inactive elements

Borders & Dividers:

- Default: Light Gray (#E0E0E0) - Standard borders
- Subtle: Very Light Gray (#F5F5F5) - Soft dividers
- Focus: Primary Blue (#4A90E2) - Active input borders

Gradients (Used Sparingly)

Hero Card Gradients:

- Success: Linear gradient from #66BB6A to #4CAF50 (healthy budget)
- Warning: Linear gradient from #FFA726 to #FB8C00 (approaching limit)
- Danger: Linear gradient from #EF5350 to #F44336 (over budget)

Subtle Background Gradients:

- Card depth: Very subtle gradient from white to #FAFBFC (top to bottom)

Color Usage Guidelines

Accessibility:

- All text meets WCAG 2.1 Level AA contrast requirements (4.5:1 minimum)
- Color-blind friendly combinations tested with simulators
- Never rely on color alone for critical information (use icons + text)

Consistency:

- Income: Always green (#66BB6A)
- Expenses: Always red/coral (#EF5350)
- Transfers: Always blue (#4A90E2)
- Budget status colors consistent across all views

Emotional Tone:

- Soft, muted tones reduce financial anxiety
- Avoid harsh reds and aggressive oranges
- Use gradients to add warmth without overwhelming

Typography

Font Family

Primary Font:

- Sans-serif system font stack:
  - iOS/macOS: SF Pro Display / SF Pro Text
  - Android: Roboto
  - Web fallback: "Inter", -apple-system, BlinkMacSystemFont, "Segoe UI", sans-serif

Monospace (for amounts):

- SF Mono (iOS), Roboto Mono (Android), "Fira Code", Consolas, monospace

Type Scale

Display (Hero Elements):

- Free to Spend amount: 48px / 3rem, Bold (700)
- Usage: Only for primary hero numbers

Heading 1:

- Size: 32px / 2rem, Bold (700)
- Usage: Screen titles, major section headers

Heading 2:

- Size: 24px / 1.5rem, Semibold (600)
- Usage: Card titles, modal headers, subsection titles

Heading 3:

- Size: 20px / 1.25rem, Semibold (600)
- Usage: List headers, category names

Body Large:

- Size: 18px / 1.125rem, Regular (400)
- Usage: Important body text, transaction amounts

Body:

- Size: 16px / 1rem, Regular (400)
- Usage: Default text, descriptions, paragraphs

Body Small:

- Size: 14px / 0.875rem, Regular (400)
- Usage: Secondary information, labels, helper text

Caption:

- Size: 12px / 0.75rem, Regular (400)
- Usage: Timestamps, metadata, footnotes

Line Height

- Display: 1.2
- Headings: 1.3
- Body: 1.5
- Captions: 1.4

Font Weights

- Regular: 400 (default text)
- Medium: 500 (emphasis, tabs)
- Semibold: 600 (headings, buttons)
- Bold: 700 (display, strong emphasis)

Spacing System

Base Unit: 4px

- 4px (0.25rem): Minimal spacing, tight groups

- 8px (0.5rem): Related elements (icon + label)
- 12px (0.75rem): Card padding (small), list item padding
- 16px (1rem): Standard padding, section spacing
- 24px (1.5rem): Between major sections
- 32px (2rem): Large section breaks
- 48px (3rem): Screen padding (top/bottom on mobile)
- 64px (4rem): Extra large spacing (rarely used)

Component Spacing:

- Button padding: 12px vertical, 24px horizontal
- Card padding: 16px (mobile), 24px (desktop)
- Input padding: 12px
- List item padding: 16px vertical, 16px horizontal

Icons & Illustrations

Icon System:

- Lucide Icons (React) for consistency
- Size variants: 16px (small), 24px (default), 32px (large), 48px (extra large)
- Stroke width: 2px (default), 1.5px (thin variant for large icons)
- Colors: Inherit from context or use palette colors

Custom Category Icons:

- Allow users to select from emoji or icon library
- Fallback to default icon set if custom not set

Illustrations:

- Minimal, friendly style for empty states
- Use primary color palette
- Optional for MVP (can use simple icons instead)

Animation & Motion

Design Philosophy:

- Purposeful, not decorative
- Reinforces spatial relationships
- Provides feedback for actions

- Never blocks user (can be skipped/interrupted)

Timing:

- Instant: <100ms - Hover states, ripples
- Fast: 150-200ms - Toggles, checkboxes, button presses
- Standard: 250-350ms - Page transitions, modal appearances
- Slow: 400-500ms - Complex animations, loading states

Easing:

- Ease-out: Default for most UI (elements entering screen)
- Ease-in: Elements leaving screen
- Ease-in-out: State changes, toggles

Key Animations:

1. **Free to Spend Update:**
    - Number counts up/down with easing
    - Background gradient subtly shifts if crossing threshold
    - Duration: 500ms

2. **Transaction Entry Modal:**
    - Slides up from bottom (mobile): ease-out, 300ms
    - Step transitions: Fade + slight vertical shift, 250ms
    - Dismissal: Slide down, ease-in, 250ms

3. **Wallet Balance Change:**
    - Old number fades out, new fades in with slight scale
    - Duration: 300ms

4. **Project Switch:**
    - Brief fade out → content swap → fade in
    - Total duration: 400ms (200ms each phase)

5. **Swipe Navigation:**
    - Screen follows finger at 1:1
    - Completes with ease-out when threshold passed
    - Snaps back with ease-in-out if canceled

6. **Card Interactions:**
    - Hover: Scale 1.02, shadow increase, 150ms
    - Press: Scale 0.98, 100ms
    - Ripple effect on tap (Material Design style)

7. **Loading States:**
    - Skeleton screens with shimmer animation

    - Progress bars with indeterminate animation
    - Spinners only for critical waits

Reduced Motion:

- Respect prefers-reduced-motion system setting
- Disable all non-essential animations
- Keep instant feedback (button presses) but remove transitions

Responsive Design

Breakpoints:

- Mobile Small: 320px - 374px
- Mobile Medium: 375px - 424px
- Mobile Large: 425px - 767px
- Tablet: 768px - 1023px
- Desktop Small: 1024px - 1439px
- Desktop Large: 1440px+

Layout Adaptations:

320px - 767px (Mobile):

- Single column layouts
- FAB navigation
- Category grid: 2 columns
- Horizontal scrolling for wallets (if >3)
- Bottom sheets for modals
- Touch-optimized spacing (min 44x44px tap targets)

768px - 1023px (Tablet):

- 2-3 column layouts where appropriate
- Category grid: 3 columns
- Larger modals (centered, max-width: 600px)
- Can show sidebar navigation or FAB (user choice)

1024px+ (Desktop):

- Sidebar navigation
- Category grid: 3-4 columns
- Multi-column dashboard (2-3 columns)
- Centered content (max-width: 1200px)

- Hover states more prominent
- Keyboard navigation support

Interaction Patterns

Touch Targets:

- Minimum size: 44x44px (iOS HIG) / 48x48px (Material Design)
- Spacing between targets: At least 8px

Feedback:

- Visual: Color change, scale, shadow
- Haptic: Gentle vibration on important actions
- Audio: Optional success sounds (off by default)

Gestures:

- Tap: Primary action
- Long press: Secondary menu (FAB)
- Swipe horizontal: Navigate between main screens
- Swipe on list item: Edit/Delete actions
- Pull to refresh: Sync data (on main screens)

Loading Patterns:

- Optimistic UI: Show changes immediately, sync in background
- Skeleton screens: For initial loads
- Progress indicators: For determinate processes
- Spinners: Only for blocking operations

Error Handling:

- Inline validation: Show errors near relevant fields
- Toast messages: For non-critical errors (auto-dismiss)
- Modal alerts: For critical errors requiring action
- Error states: Clear recovery actions provided

Design System Summary

Component Library (To Be Built):

- Buttons: Primary, Secondary, Tertiary, Destructive, Icon
- Inputs: Text, Number, Date, Search, Dropdown
- Cards: Standard, Wallet, Category, Summary, Transaction
- Lists: Standard, Swipeable, Grouped
- Modals: Full-screen (mobile), Centered (desktop), Bottom sheet
- Navigation: FAB, Radial menu, Sidebar, Tabs
- Feedback: Toast, Alert, Confirmation dialog, Loading states
- Data Display: Progress bars, Charts (simple), Stats cards, Badges

Consistency Guidelines:

- Reuse components across screens
- Maintain consistent spacing using 4px base unit
- Keep color usage semantic and consistent
- Use same interaction patterns throughout
- Match system conventions (iOS/Android/Web)

Accessibility Considerations

Color Contrast:

- All text meets WCAG 2.1 Level AA (4.5:1 for normal text, 3:1 for large text)
- Interactive elements meet 3:1 contrast with background

Screen Readers:

- Semantic HTML elements
- ARIA labels for icons and interactive elements
- Descriptive alt text for images/illustrations
- Logical heading hierarchy
- Focus indicators clearly visible

Keyboard Navigation:

- All interactive elements keyboard accessible
- Logical tab order
- Escape key closes modals
- Enter/Space activates buttons

Motion & Animation:

- Respect prefers-reduced-motion
- No auto-playing animations >5 seconds
- No flashing content (epilepsy risk)

Touch Targets:

- Minimum 44x44px
- Adequate spacing between targets
- No critical actions require precise tapping

Text:

- Resizable text support (up to 200%)
- No text in images (unless decorative)
- Clear, simple language
- Avoid jargon where possible

Future Design Enhancements (Post-MVP)

- Dark mode (auto/manual toggle)
- Theme customization (accent colors)
- Advanced data visualizations (charts, graphs)
- Widget support (iOS/Android home screen)
- Apple Watch / Wear OS companion apps
- Tablet-optimized layouts with split views
- Advanced animations and micro-interactions
- Personalized dashboard layouts
- Custom icon packs for categories/wallets

8. User Flows

Flow 1: First-Time User Onboarding

Phase 1: Quick Setup (3-4 minutes)

1. User opens app → Welcome screen with value proposition

- Tagline: "Your Financial Breathing Space"
- Value props: Automatic tracking, Offline-first, See what you can spend guilt-free
- "Get Started" button

2. Sign-Up/Sign-In Screen

- Header: "Welcome to Headroom"
- **New Users:**
    - "Continue with Google" button (OAuth)
    - "Continue with Apple" button (OAuth)
    - "Continue with Microsoft" button (OAuth)
    - Divider: "or"
    - "Sign up with Email" button
- **Returning Users:**
    - "Already have an account? Sign in" link
    - Leads to sign-in screen with email/password + OAuth options
- On successful authentication → System automatically creates default "Personal" project in background

3. Project & Wallet Setup (4 screens):

Screen 1: "Choose your currency"

- Context: "What currency do you use?"
- List of common currencies with flags and symbols:
    - USD - United States Dollar ($)
    - UGX - Ugandan Shilling (USh)
    - EUR - Euro (€)
    - GBP - British Pound (£)
    - KES - Kenyan Shilling (KSh)
    - [More currencies...]
- Search bar for finding specific currency
- "Continue" button

Screen 2: "Let's set up your first wallet"

- Context: "Where do you keep your money? We'll track these separately."

- Quick-add wallet cards:
    - Cash
    - Mobile Money
    - Bank Account
    - Credit Card
    - Custom
- User selects one or more (can skip and add later)
- For each selected wallet, inline input appears: "Starting balance" (optional, can be $0)
- "Continue" button (enabled after at least one wallet added)

Screen 2: "Enter your monthly income"

- Large input field with currency symbol
- Helper text: "This helps us calculate how much you can spend"
- Optional: "Which wallet does this go to?" dropdown (defaults to first wallet created)
- Can skip if income is irregular
- "Next" button

Screen 4: "Add your recurring bills (optional)"

- Quick-add interface:
    - Bill name input (e.g., "Rent")
    - Amount input
    - Recurrence period dropdown: Daily / Weekly / Monthly / Yearly (default: Monthly)
    - "Add Bill" button
- Shows list of added bills with amounts and recurrence
- Helper text: "You can always add more later"
- "Skip" or "Continue" button

Screen 5: "Choose your tracking period"

- Three large option cards:
    - **Daily** - "Reset every day"
    - **Weekly** - "Reset every week" (recommended for frequent spenders)
    - **Monthly** - "Reset every month" (most popular)
- Helper text: "Your Free to Spend amount will reset each [period]. Unspent money rolls over automatically."
- Selection highlights the card
- "Finish Setup" button

5. Success screen: "Setup complete!"

- Shows calculated Free to Spend amount: "$X available this [period]"
- "Your Personal project is ready"
- "Start Tracking" button

Phase 2: Interactive Tutorial (2-3 minutes)

Tutorial Mode Activated - Overlay guides with actual UI

Step 1: Dashboard Introduction

- Spotlight on Free to Spend hero number
- Overlay: "This is your Free to Spend amount for this [period] in your Personal project"
- Shows wallet balance cards below
- "Got it" to continue

Step 2: Project Context

- Spotlight on header showing "Personal" project name
- Overlay: "You're in your Personal project. You can create more projects later for different purposes (business, savings, etc.)"
- "Next" to continue

Step 3: Adding First Transaction

- Spotlight on FAB button
- Overlay: "Tap here to log your first transaction"
- User taps → Transaction flow begins (guided)
- User actually adds a real transaction through full flow
- After save, tutorial resumes

Step 4: Category Navigation

- Overlay: "Swipe left or tap a category to see spending by category"
- User swipes/taps to category view
- Shows their first transaction in the category
- "This is where you'll track spending by category"
- "Next" to continue

Step 5: Quick Navigation

- Spotlight on FAB
- Overlay: "Hold this button to quickly navigate between Dashboard, Categories, and Debt Tracker"
- User tries long-press → Radial menu appears
- "Perfect!" confirmation

Step 6: Completion

- "You're all set! Log transactions throughout the day to stay on track."
- "Start Using Headroom" button

- Dashboard loads with real data

Phase 3: Extended Setup (Delayed to Week 1)

After 3 days of usage → In-app prompt:

- *"Let's enhance your tracking"*
- *Optional deeper setup options (can skip any/all):*

1. Add More Wallets

- Quick-add interface for additional wallets
- Set starting balances

2. Create Category Budgets

- Shows spending summary for past 3 days by category
- "Set monthly budget for Food: $___"
- Can set budgets for multiple categories

3. Add Existing Debts

- "Track loans and debts"
- Name, total amount, minimum payment
- Links to debt tracker

4. Customize Categories

- "Remove unused categories?"
- Shows categories with 0 transactions
- Option to hide/remove
- "Add custom categories" option

5. Configure Notifications

- Daily spending reminders (time picker)
- Weekly summaries (day/time picker)
- Budget alerts (toggle on/off)

Can skip any step and revisit in Settings → Complete or "Remind me later"

Flow 2: Logging a Transaction (Conversational, Sequential with Wallets)

Design Philosophy: Feels like a conversation, not a form. Only one section active at a time, others fade into background. Project context is maintained (already selected).

1. User taps FAB → Screen slides up smoothly

- Header shows current project name in small text: "Personal"
- Background slightly dims

2. Step 1: Transaction Type (ACTIVE - full color, large)

- Header: "What type of transaction?"
- Three large, touch-friendly cards:
    - **Income** - "Money received"
    - **Expense** - "Money spent"
    - **Transfer** - "Move between wallets"
- User taps choice → Card animates with checkmark
- Next step slides up smoothly
- Previous step fades to 30% opacity, moves up slightly

3A. If INCOME selected:

Step 2: Amount (ACTIVE)

- Header: "How much did you receive?"
- Large numeric keypad (optimized for thumb typing)
- Amount displays large as user types: "$___"
- Currency symbol based on project settings
- "Next" button or swipe up

Step 3: Destination Wallet (ACTIVE)

- Header: "Which wallet?"
- List of project's wallets with current balances:
    - Cash ($125.50)
    - Mobile Money ($450.00)
    - Bank Account ($2,340.75)
- Default wallet (if set) is pre-selected but can change
- User taps wallet → Next step slides in

Step 4: Category (ACTIVE)

- Header: "Category"
- Grid of category icons + labels (income categories):

    - Salary
    - Gift
    - Refund
    - Investment
    - Other
    - **+ Add Custom**
- User taps category → Next step slides in

Step 5: Recurring? (ACTIVE)

- Header: "Does this repeat?"
- Toggle: "Make this recurring"
- If toggled ON, inline fields appear:
    - Frequency: Daily / Weekly / Monthly / Yearly
    - Start date: Date picker (defaults to today)
    - End date: Date picker (optional)
    - Confirmation: Toggle "Ask before adding each time"
- "Next" or "Skip"

Step 6: Notes (Optional) (ACTIVE)

- Header: "Add a note (optional)"
- Single text input field
- Placeholder: "E.g., November paycheck"
- "Skip" or "Next"

Step 7: Review & Save (ACTIVE)

- Summary card:
    - (checkmark) Income: $XXX
    - To: [Wallet name]
    - Category: [Category]
    - [Recurring badge if applicable]
    - Note: [if added]
- Large "Save Transaction" button
- Small "Edit" link if user wants to change something

Completion:

- Screen slides down
- Returns to dashboard
- Wallet balance updates with animation
- Free to Spend recalculates
- Success toast: "Income added ✓"
- Haptic feedback

3B. If EXPENSE selected:

Step 2: Amount (ACTIVE)

- Same as income

Step 3: Source Wallet (ACTIVE)

- Header: "Pay from which wallet?"
- List of project's wallets with current balances
- Visual indicator if wallet has insufficient balance (red text)
- User taps wallet → Next step

Step 4: Category (ACTIVE)

- Header: "What's this for?"
- Grid of expense categories:
    - Food
    - Transport
    - Shopping
    - Bills
    - Entertainment
    - Health
    - Other
    - **+ Add Custom**
- User taps category → Next step

Step 5-7: Same as Income flow (Recurring, Notes, Review)

3C. If TRANSFER selected:

Step 2: Amount (ACTIVE)

- Header: "How much to transfer?"
- Large numeric keypad
- Amount displays: "$___"
- "Next"

Step 3: Source Wallet (ACTIVE)

- Header: "Transfer from"
- List of wallets
- User selects → Next step

Step 4: Destination Wallet (ACTIVE)

- Header: "Transfer to"
- List of wallets (excludes source wallet)
- User selects → Next step

Step 5: Transfer Fee (Optional) (ACTIVE)

- Header: "Any transfer fee?"
- Toggle: "Include transfer fee"
- If ON: Amount input appears
- Helper text: "Fee will be deducted from destination"
- "Skip" or "Next"

Step 6: Notes (Optional) (ACTIVE)

- Header: "Add a note (optional)"
- Placeholder: "E.g., Cash withdrawal"
- "Skip" or "Next"

Step 7: Review & Save (ACTIVE)

- Summary card:
    - (transfer icon) Transfer: $XXX
    - From: [Source wallet] → To: [Destination wallet]
    - Fee: $XX (if applicable)
    - Note: [if added]
- "Complete Transfer" button

Completion:

- Screen slides down
- Both wallet balances update with animation
- Toast: "Transfer complete ✓"
- Haptic feedback

Key UX Principles:

- Sequential = less cognitive load, focus on one thing at a time
- Project context maintained throughout (shown in header, not selected again)
- Default wallet speeds up common flows
- Previous steps visible but faded = context without distraction
- Swipe up between steps = feels fluid and modern
- Skip/back options always available but not prominent
- Fast path for common scenario: Expense → Amount → Wallet → Category → Done (4 taps + typing)

Flow 3: Switching Projects

Entry Points:

1. *Tap project name in header*
2. *Long-press on project name*
3. *Settings → Manage Projects*

Flow:

1. Tap project name in header → Dropdown appears

Dropdown Contents:

- Current project highlighted with checkmark:
    - (checkmark) **Personal** (Current)
    - Current balances: 3 wallets, $2,915.75 total
- Other projects listed:
    - **Side Business**
    - 2 wallets, $1,450.00 total
    - **Family Savings**
    - 1 wallet, $3,200.00 total
- Divider line
- **+ Create New Project** button
- (settings icon) **Manage Projects** button (leads to full management)

2. User taps different project → Instant switch

- Dropdown closes
- Loading indicator (brief, <1 second)
- Entire UI updates:
    - Header shows new project name
    - Dashboard recalculates for new project
    - Wallet cards update
    - Free to Spend recalculates
    - All data now scoped to selected project
- Smooth transition animation
- Toast: "Switched to [Project Name]"

3. User taps "+ Create New Project"

- Modal slides up: "Create New Project"
- Input: "Project name"
- Suggested names: Work, Business, Savings, Travel
- "Create" button
- On create:

    - New project created
    - Automatically switches to new project
    - Onboarding prompt: "Set up your first wallet for [Project Name]"
    - Quick wallet creation flow (same as onboarding)

4. User taps "Manage Projects" → Full management screen

Manage Projects Screen:

- List of all projects with cards:
    - **Personal** (star icon) (Primary badge)
        - 3 wallets, $2,915.75 total
        - Last used: Today
        - Actions: Edit | Set as Primary | Delete
    - **Side Business**
        - 2 wallets, $1,450.00 total
        - Last used: 2 days ago
        - Actions: Edit | Set as Primary | Delete
    - **Family Savings**
        - 1 wallet, $3,200.00 total
        - Last used: 1 week ago
        - Actions: Edit | Set as Primary | Delete
- **+ Create New Project** button at bottom

Edit Project:

- Taps Edit → Modal: "Edit Project Name"
- Input with current name
- "Save" button
- Updates project name everywhere

Set as Primary:

- Taps "Set as Primary" → Confirmation
- "Make [Project Name] your primary project? This will load automatically when you open the app."
- "Confirm" | "Cancel"
- On confirm: Star badge moves to new primary

Delete Project:

- Taps Delete → Warning modal
- "Delete [Project Name]?"
- "This will permanently delete all wallets, categories, transactions, budgets, and debts in this project. This cannot be undone."
- Text input: "Type DELETE to confirm"

- "Delete Project" button (disabled until correct text entered)
- "Cancel" button
- On delete:
    - Project removed
    - If was active project, switches to primary or first available
    - Toast: "[Project Name] deleted"

Flow 4: Managing Wallets

Flow 4A: Creating a Wallet

Entry Points:

1. *Dashboard → "+" on wallet section*
2. *Settings → Wallets → Add Wallet*
3. *During transaction if no wallets exist*

Flow:

1. Tap "+ Add Wallet" → Modal slides up

Screen: "Add New Wallet"

- Header: "Add wallet to [Project Name]"
- Quick-add preset cards:
    - Cash
    - Mobile Money
    - Bank Account
    - Credit Card
    - Other
- **+ Custom** button

2. User selects preset (e.g., "Cash")

- Name field pre-filled: "Cash" (editable)
- Starting balance input: "$___" (optional, defaults to $0)
- Helper text: "Enter your current balance in this wallet"
- Color picker (optional): Select icon color
- "Add Wallet" button

3. Completion

- Modal closes
- New wallet appears in wallet list

- If on dashboard, wallet card animates in
- Toast: "Cash wallet added ✓"

Flow 4B: Transferring Between Wallets

Entry Points:

1. *FAB → Select "Transfer" type (already covered in Flow 2)*
2. *Wallet card → "Transfer" button*
3. *Wallet detail view → "Transfer" button*

Flow from Wallet Card:

1. Tap wallet card on dashboard → Quick actions appear

- View Details
- Transfer
- Edit
- Transactions

2. Tap "Transfer" → Transfer flow (same as Flow 2, Step 3C)

- Source wallet pre-selected (the one user tapped)
- Proceeds through transfer steps

Flow 4C: Editing Wallet Balance

Use Case: *User's actual wallet balance differs from app balance (e.g., found extra cash, forgot to log transaction)*

Entry Points:

1. *Wallet card → "Edit"*
2. *Wallet detail view → "Adjust Balance"*

Flow:

1. Tap "Edit" on wallet card → Modal appears

Screen: "Adjust Wallet Balance"

- Current balance shown: "$125.50"
- Input: "New balance: $___"

- Helper text: "This will create an adjustment transaction"
- Reason dropdown (optional):
    - Found money
    - Forgot to log expense
    - Correction
    - Other
- Notes field (optional)
- "Update Balance" button

2. User enters new balance → Confirmation

- Shows difference: "+$20.00" or "-$15.00"
- "This will add an adjustment transaction to your records"
- "Confirm" | "Cancel"

3. On confirm:

- Adjustment transaction created
- Wallet balance updates
- Free to Spend recalculates
- Toast: "Balance adjusted ✓"

Flow 4D: Viewing Wallet Details

1. Tap wallet card → Wallet Detail Screen

Screen: "[Wallet Name] Details"

- Header: Wallet name, balance (large)
- Quick actions bar:
    - Transfer
    - Adjust Balance
    - Settings
- Transaction history for this wallet:
    - Grouped by date
    - Shows: amount, category, type (income/expense/transfer)
    - Swipe actions: Edit | Delete
- Filter options: Date range, Type, Category
- "View All Transactions" button (opens Reports with wallet filter applied)

Wallet Settings (via settings icon):

- Rename wallet
- Change icon/color

- Set as default wallet for project
- Archive wallet (hides from active wallets but preserves history)
- Delete wallet (requires confirmation, only if balance is $0)

Flow 5: Navigating by Category

Primary Navigation Pattern: *Categories replace time-based transactions as main view*

Flow 5A: Viewing Category Dashboard

1. From Dashboard → Swipe left OR tap "Categories" in navigation

Screen: "Categories" (shows all categories for active project)

Layout:

- Header: "[Project Name] - Categories"
- Grid of category cards (2 columns on mobile):
    - **Food** (icon)
        - Spent: $245.80 / $500 (budget)
        - Progress bar (49% - green)
        - Last transaction: 2 hours ago
    - **Transport** (icon)
        - Spent: $120.00 / $200
        - Progress bar (60% - yellow)
        - Last transaction: Yesterday
    - **Shopping** (icon)
        - Spent: $340.50 / $300
        - Progress bar (113% - red)
        - (warning icon) Over budget
    - [Additional categories...]
    - **+ Add Category** card
- FAB available: Tap to add transaction (will ask for category during flow)

2. Tap on a category card → Category Detail View

Flow 5B: Category Detail View

Screen: "Food - [Project Name]"

Top Section:

- Large spending summary:
    - "$245.80 spent this month"
    - Budget: $500
    - Progress bar with percentage
    - "Free to spend in Food: $254.20"
- Quick stats:
    - Average per transaction: $15.36
    - Number of transactions: 16
    - Biggest expense: $45.00 (Restaurant, Dec 10)

Transaction List:

- All transactions in this category
- Grouped by date (Today, Yesterday, This Week, Earlier)
- Each transaction shows:
    - Amount, description/note, wallet used, time
    - Swipe left: Edit | Delete
    - Tap: Transaction details

Actions:

- FAB: "Add [Category] Transaction" (pre-fills category in flow)
- Top right: (settings icon) → Category Settings
    - Set/Edit Budget
    - Rename Category (if custom)
    - Change Icon/Color
    - Hide Category
    - Delete Category (only if custom and no transactions)

3. Tap "Add [Category] Transaction"

- Opens transaction flow (Flow 2)
- Category pre-selected (skips category step)
- Proceeds: Amount → Wallet → Recurring → Notes → Save
- Returns to category view with new transaction visible

Flow 5C: Adding Custom Category

1. From Categories view → Tap "+ Add Category" card

Modal: "Create Category"

- Category name input
- Icon picker (emoji or icon library)
- Color picker
- Budget (optional): "$___/month"
- "Create Category" button

2. On create:

- New category appears in grid
- Alphabetically sorted (or user can reorder via Settings)
- Toast: "[Category Name] created ✓"
- Can immediately tap to add first transaction

Flow 5D: Category Cleanup (Week 1 Prompt)

After 1 week of use → In-app notification:

- "We noticed a few unused categories in this project. Would you like to clean them up?"
- Shows unused standard categories:
    - "You haven't used Entertainment (0 transactions)"
    - "You haven't used Health (0 transactions)"
- Options for each:
    - Keep (in case user plans to use)
    - Hide (can unhide in Settings)
- "Done" button

Flow 6: Editing a Recurring Transaction

Context: *All recurring transactions are project-scoped*

Entry Points:

1. *From category view → Tap transaction with recurring badge*
2. *From Reports → Tap transaction with recurring badge*
3. *From wallet details → Tap recurring transaction*

Flow:

1. Tap on recurring transaction → Transaction Details modal

Screen: Transaction Details

- Amount: $1,500
- Type: Income
- Category: Salary
- Wallet: Bank Account
- Date: Dec 1, 2024
- (recurring icon) **Recurring badge** (prominent)
    - "Repeats monthly on the 1st"
    - "Started: Jan 1, 2024"
    - "Next: Jan 1, 2025"
- Note: "Monthly paycheck"
- Actions:
    - Edit This Transaction
    - Edit Recurring Pattern
    - Delete

2. Tap "Edit Recurring Pattern" → Edit Recurring Modal

Screen: "Edit Recurring Transaction"

- All recurring fields editable:
    - Amount: $1,500 (editable)
    - Frequency: Monthly (dropdown: Daily/Weekly/Monthly/Yearly)
    - Start date: Jan 1, 2024 (date picker)
    - End date: None (optional date picker)
    - Confirmation: Toggle "Ask before adding"
- Toggle: "Pause recurring" (stops future generation without deleting)
- Apply changes to:
    - Radio: **This occurrence only** (one-time edit)
    - Radio: **All future occurrences** (updates pattern going forward)
- "Save Changes" button
- "Delete Recurring Pattern" button (red, bottom)

3A. User selects "All future occurrences" and saves:

- Confirmation: "Update all future [Category] transactions to $XXX [frequency]?"
- "Confirm" | "Cancel"
- On confirm:
    - Recurring pattern updated
    - Future transactions will use new parameters
    - Already-generated future transactions updated

    - Past transactions unchanged
    - Modal closes
    - Free to Spend recalculates if amount changed
    - Toast: "Recurring transaction updated ✓"

3B. User taps "Delete Recurring Pattern":

- Warning modal: "Delete recurring pattern?"
- "This will stop generating future transactions. Past transactions will remain."
- Options:
    - "Delete future only" (keeps pattern, stops generating)
    - "Delete pattern and all future transactions" (removes pattern + deletes future entries)
- "Cancel" button
- On confirm:
    - Pattern deleted/paused
    - Future transactions handled per selection
    - Toast: "Recurring transaction stopped ✓"

Flow 7: Accessing Reports & Time-Based Views

Context: *Since categories are primary navigation, chronological view moved to Reports*

Entry Points:

1. *Main navigation/menu → "Reports"*
2. *Settings → Reports*
3. *Category view → "View All Transactions" (filtered by category)*

Flow:

1. Tap "Reports" → Reports Dashboard

Screen: "Reports - [Project Name]"

Top Section - Summary Cards:

- Time period selector: This Month (dropdown icon) (dropdown: Today, Week, Month, Quarter, Year, Custom)
- Summary for selected period:
    - Total Income: $3,500
    - Total Expenses: $2,340
    - Net: +$1,160 (green)
    - Savings Rate: 33%

Insights Section:

- "Biggest expense: $450 - Rent (Bills)"
- "Most frequent category: Food (23 transactions)"
- "Busiest day: Friday (8 transactions)"
- "Most used wallet: Mobile Money (45% of expenses)"

Transaction List:

- All transactions for active project in chronological order
- Grouped by date (Today, Yesterday, This Week, Dec 10, Dec 9...)
- Each transaction:
    - Icon (income/expense/transfer)
    - Amount (color-coded: green income, red expense, blue transfer)
    - Category, Wallet
    - Description/note if exists
    - Time
- Swipe actions: Edit | Delete
- Tap: Transaction details

Filter Bar (top):

- (search icon) Search transactions
- (calendar icon) Date range picker
- (folder icon) Category filter (multi-select)
- (wallet icon) Wallet filter (multi-select)
- (currency icon) Amount range
- (filter icon) Type: All / Income / Expense / Transfer

Actions:

- Top right: Export button → "Export as CSV"
- FAB: Add Transaction (opens Flow 2)

2. Apply filters → Results update in real-time

- Transaction count updates: "Showing 34 of 156 transactions"
- Summary cards recalculate for filtered set
- Can clear all filters with "Clear Filters" button

3. Export Transactions:

- Tap Export → Modal: "Export Transactions"
- Options:
    - Time period: Current filter / Custom range
    - Format: CSV (Excel-compatible)

    - Include: All fields / Basic (date, amount, category)
- "Export" button
- File downloads/shares via system share sheet
- Toast: "Transactions exported ✓"

Additional Micro-Flows

Micro-Flow: Setting Category Budget

From Category Detail View → Tap "⚙️" → Set Budget

Modal: "Set Budget for [Category]"

- Current spending this month: $245.80
- Budget amount: "$___/month"
- Helper: Suggested based on last 3 months average (if data exists)
- Budget period: Monthly (for MVP, future: Weekly/Daily)
- Alert preferences:
    - Notify at 80% (toggle)
    - Notify at 100% (toggle)
    - Notify when exceeded (toggle)
- "Save Budget" button

On save:

- Budget stored for category in active project
- Progress bar now visible in category views
- Free to Spend calculation includes this commitment
- Toast: "Budget set for [Category] ✓"

Micro-Flow: First Transaction in New Project

When user creates new project and switches to it:

Empty State (Dashboard):

- Hero text: "Welcome to [Project Name]!"
- "You haven't added any wallets yet"
- Large "Add Your First Wallet" button
- Helpful tip: "Start by adding where you keep money (Cash, Bank, etc.)"

After adding wallet:

- Dashboard updates with wallet card
- Free to Spend shows wallet balance
- Prompt: "Ready to log your first transaction?"
- FAB pulses gently (visual cue)

Micro-Flow: Handling Insufficient Balance

During expense transaction (Flow 2):

If user selects wallet with balance < expense amount:

- Warning appears inline (Step 3 - Source Wallet)
- Selected wallet shows:
    - Credit Card ($50.00)
    - "Insufficient balance" (yellow/red text)
- User can still proceed (might be intentional overdraft/credit)
- Or user can tap "Choose different wallet" to switch

On save with insufficient balance:

- Confirmation modal:
    - "This transaction ($75) exceeds your Credit Card balance ($50)"
    - "Your balance will become -$25"
- "Proceed Anyway" | "Cancel"
- User confirms → Transaction saves, balance goes negative (shown in red)

Key Flow Principles Across All Flows:

- **Project context always visible** (header shows current project)
- **Wallet abstraction** (cards visible, but not requiring constant interaction)
- **Category-first navigation** (replaces time-based as primary view)
- **Sequential, conversational flows** (one decision at a time)
- **Smart defaults** (remembers project, suggests default wallet)
- **Progressive disclosure** (advanced options hidden until needed)
- **Immediate feedback** (animations, toasts, balance updates)
- **Forgiving UX** (easy to edit, undo, adjust)
- **Consistent patterns** (similar flows feel familiar across features)

9. Open Questions

Critical Questions (Need Resolution Before Development Starts)

1. Categories: ✅ RESOLVED

Question: Should we use predefined categories or allow users to create custom ones?

Resolution:

- Predefined standard categories (Food, Transport, Shopping, Bills, Entertainment, Health, Other) exist in every project
- Users can add unlimited custom categories per project
- Standard categories can be hidden (not deleted) if unused
- Week 1 cleanup prompt suggests hiding unused standard categories

2. Currency Support in Multi-Project Scenario

Question: Should users be able to have different currencies in different projects?

Current Implementation:

- User sets a default currency during onboarding
- Each project can theoretically have its own currency (field exists in schema)
- Free to Spend calculation assumes single currency

Open Considerations:

- **Option A:** Lock all projects to user's default currency (simpler for MVP)
- **Option B:** Allow per-project currency but don't convert between them
- **Option C:** Allow per-project currency with conversion rates (complex, post-MVP)

Recommendation: Option A for MVP - Lock all projects to user's default currency. Add multi-currency in Phase 2.

Impact: Low - Most users will operate in single currency initially

3. Data Retention Policy

Question: How long should we keep historical transaction data?

Considerations:

- Users may want years of history for tax/legal purposes
- Storage costs scale with data volume
- Performance may degrade with large datasets

Options:

- **Option A:** Unlimited storage (no deletion) with pagination
- **Option B:** Rolling window (e.g., keep last 2 years, archive older)
- **Option C:** Let users choose retention period in settings

Recommendation: Option A for MVP - No automatic deletion. Users can manually delete old projects/transactions if needed. Revisit if storage becomes expensive.

Impact: Medium - Affects database design and long-term costs

4. Sharing & Collaboration

Question: Should users be able to share projects with family members or partners?

Use Cases:

- Couples managing joint finances
- Families tracking shared expenses
- Business partners managing business project together

Current State: Each project is owned by single user (user_id foreign key)

Options:

- **Option A:** No sharing in MVP (current approach)
- **Option B:** Read-only sharing (others can view, not edit)
- **Option C:** Full collaboration (multiple users can edit)

Implications:

- Option B/C require: Project members table, permission system, conflict resolution for simultaneous edits
- Option C adds significant complexity to sync and RLS policies

Recommendation: Option A for MVP - No sharing. Add as Phase 2 feature based on user demand.

Impact: High complexity if implemented, but not critical for MVP

5. Export Formats

Question: What export formats should we support for user data?

User Needs:

- Backup personal data
- Tax preparation
- Migration to other tools
- Sharing with accountant

Options:

- **CSV** (transactions only) - Simple, universally compatible
- **Excel** (.xlsx) - Formatted, multiple sheets for different data types
- **PDF** - Visual reports, not editable
- **JSON** - Complete data dump, developer-friendly

Recommendation: Start with CSV (transactions only) in MVP. Add Excel and PDF reports in Phase 2.

Implementation:

- Week 7: Basic CSV export of transactions with filters
- Post-MVP: Excel export with multiple sheets (transactions, budgets, debts)
- Post-MVP: PDF reports with charts and summaries

Impact: Low - CSV is sufficient for most MVP users

6. Notification Frequency & Timing

Question: What's the optimal frequency and timing for insights without annoying users?

Current Plan:

- Daily spending reminder: 7-8 AM
- Weekly summary: Sunday evening 6-8 PM
- Budget alerts: As they happen (80%, 100%, 120%)
- Maximum 2 notifications per day (excluding critical alerts)

Open Considerations:

- Should notification times be customizable per user?
- Should we have notification presets? (Morning Person, Night Owl, Minimal)
- Should we use AI/ML to learn optimal timing per user? (future)

Options:

- **Option A:** Fixed times for all users (7-8 AM, 6-8 PM)
- **Option B:** User-customizable times during onboarding
- **Option C:** User-customizable times anytime in settings

Recommendation: Option C - Allow customization in settings, with smart defaults (7-8 AM, 6-8 PM). Users who care will adjust.

Impact: Low - Adds settings UI but not complex logic

7. Wallet Balance Calculation Method

Question: Should wallet balances be calculated from transactions or manually set and tracked?

Current Implementation:

- Users set initial balance when creating wallet
- Transactions automatically adjust balance
- Users can manually adjust if balance drifts from reality

Open Consideration:

- What happens if user's real-world balance differs from app balance?
    - They forgot to log transactions
    - Cash was lost/stolen
    - Bank fees not recorded
    - Rounding errors accumulated

Resolution:

- Provide "Adjust Balance" feature (already in design)
- Creates an "adjustment" transaction type to reconcile differences
- User can add note explaining the adjustment
- Balance history preserved for audit trail

Question: Should we prompt users to reconcile balances periodically?

Recommendation: Optional weekly reminder "Verify your wallet balances" with one-tap access to each wallet. Can be disabled in settings.

Impact: Low - Improves accuracy, minor UX addition

8. Transfer Fees: Deducted from Source or Destination?

Question: When a wallet transfer has a fee, which wallet does it come from?

Scenario: Transfer $100 from Bank to Cash with $2 ATM fee

Options:

- **Option A:** Deduct from destination ($100 leaves bank, $98 arrives in cash)
- **Option B:** Deduct from source ($102 leaves bank, $100 arrives in cash)
- **Option C:** Let user choose during transfer

Current Implementation: Transfer metadata includes fee amount, but deduction logic not fully specified.

Recommendation: Option A - Deduct from destination (most intuitive, matches ATM behavior). $100 leaves source, $100 - fee arrives at destination.

Implementation:

- Transaction record: amount = $100, transfer_fee = $2
- Source wallet: -$100
- Destination wallet: +$98
- Fee displayed in transaction detail

Impact: Low - Simple logic, clear to users

9. Recurring Transaction Confirmation Behavior

Question: How should "request confirmation before adding" recurring transactions work?

Current Design: Toggle in recurring setup asks "Ask before adding each time"

Options:

- **Option A:** Notification with Approve/Skip buttons (requires notification interaction)
- **Option B:** In-app prompt on next app open after due date (less intrusive)
- **Option C:** Transaction added to "pending" list requiring user review

Considerations:

- Option A: User might not see notification or dismiss it
- Option B: User might not open app on due date
- Option C: Adds complexity but gives user control

Recommendation: Option B for MVP - When user opens app and recurring transaction is due, show non-blocking banner: "Rent payment due today ($1,200). Confirm or skip?"

Fallback: If user doesn't open app for 3 days, auto-add transaction (user can delete if incorrect).

Impact: Medium - Affects user experience, needs thoughtful UX

10. Deleted Projects: Hard Delete or Soft Delete?

Question: When user deletes a project, should data be permanently deleted or archived?

Considerations:

- **Hard delete:** Data gone forever, smaller database, clean slate
- **Soft delete:** Data preserved with deleted_at flag, can be recovered, audit trail

Implications:

- Hard delete: Cannot undo, user loses all history
- Soft delete: Database grows over time, need cleanup policy

Recommendation: Soft delete with recovery window:

- Set deleted_at timestamp instead of removing from database
- Hide from UI immediately
- User can recover within 30 days via support/settings
- Hard delete after 30 days (automated job)

Implementation:

- Add deleted_at column to Project model
- Filter out deleted projects in queries (WHERE deleted_at IS NULL)
- Background job runs monthly to purge projects where deleted_at > 30 days ago

Impact: Low - Better user experience, minimal complexity

11. Free to Spend Period Rollover Logic

Question: How should unspent "Free to Spend" amounts roll over to the next period?

Current Design: "Unspent money automatically rolls over"

Scenario: User has $200 Free to Spend this week, only spends $150. Next week starts with base Free to Spend + $50 rollover?

Options:

- **Option A:** Cumulative (rollover adds to next period's Free to Spend)
- **Option B:** Reset (fresh calculation each period, no rollover)
- **Option C:** Optional (user chooses in settings)

Consideration:

- Option A encourages saving by making unspent money visible
- Option B is simpler to understand and calculate
- Option C gives flexibility but adds complexity

Recommendation: Option A - Cumulative rollover. It aligns with "breathing space" concept and rewards underspending.

Calculation:

Next Period Free to Spend =

(Income for period) - (Committed expenses/bills) + (Unspent from last period)

Edge Case: What if user overspent last period? Rollover is negative (reduces next period's Free to Spend).

Impact: Medium - Core to the product promise, needs to feel right

12. Category Budget Period vs Project Tracking Period

Question: Can category budgets have different periods than the project's tracking period?

Example: Project tracks daily, but user wants monthly Food budget

Current Schema: Budget has period field (daily/weekly/monthly)

Options:

- **Option A:** Budget periods must match project tracking period (simpler)
- **Option B:** Budget periods can differ (more flexible)

Recommendation: Option B - Allow different periods. A user tracking daily might still want monthly category budgets.

Implementation: Calculate budget progress based on budget's own period, not project's period.

Impact: Low - Schema already supports this, just needs proper calculation logic

13. Empty State Messaging: Encouraging vs Neutral?

Question: Should empty states encourage action or be neutral/informative?

Examples:

- **Encouraging:** "No transactions yet! Tap the + button to start tracking your spending 🎯"
- **Neutral:** "No transactions to display"

Trade-offs:

- Encouraging: More engaging, guides new users, can feel patronizing to experienced users
- Neutral: Professional, clean, doesn't assume user intent

Recommendation: Hybrid approach:

- New users (first 7 days): Encouraging messages with guidance
- Experienced users (7+ days): Neutral messages
- Can be toggled in settings: "Helpful hints" on/off

Impact: Low - UI copy decision, easily adjustable

14. Transaction Descriptions: Required or Optional?

Question: Should users be required to add a description/note to transactions?

Current Design: Description is optional in transaction flow

Considerations:

- **Required:** Better record-keeping, easier to remember transactions later
- **Optional:** Faster entry, less friction (aligns with "effortless" goal)

User Scenarios:

- Coffee purchase: Amount and category ("Food") enough? Or need "Starbucks latte"?
- Recurring salary: Description mostly redundant
- One-time large expense: Description very helpful

Recommendation: Keep optional, but encourage for transactions >$50 with prompt: "Add a note so you remember this later?"

Impact: Low - UX nudge, not technical change

15. Initial Wallet Balances: Verified or Trusted?

Question: Should we verify wallet balances match reality before user starts tracking?

Scenario: User creates "Bank Account" wallet with $5,000 initial balance. How do we ensure accuracy?

Options:

- **Option A:** Trust user input (current approach)
- **Option B:** Prompt verification: "Open your banking app and confirm your current balance"
- **Option C:** Bank integration to auto-fetch balance (future, complex)

Recommendation: Option B - Add confirmation prompt during wallet creation: "Make sure this matches your real balance to keep tracking accurate."

Plus: Provide easy "Adjust Balance" feature for corrections.

Impact: Very Low - Just UI copy change

16. Monetization Strategy (Future Consideration)

Question: How will Headroom generate revenue?

Not urgent for MVP, but good to consider architecture implications

Options:

- **Freemium:** Free tier + paid premium features
    - Premium: Advanced analytics, unlimited projects, cloud backup, priority support
- **One-time purchase:** Pay once, own forever
- **Subscription:** Monthly/annual fee for access
- **Ads:** Free with ads, pay to remove (not recommended for finance app)

Recommendation: Plan for freemium model:

- MVP: Free for all features
- Post-launch: Keep core features free forever
- Phase 2: Add premium features (advanced reports, business tools, team collaboration)

Architectural Impact:

- Add subscription_tier field to User model (future)
- Design features with tier gating in mind
- Ensure free tier remains valuable

Decision Timeline: Can wait until 6 months post-launch based on user feedback

17. Security: End-to-End Encryption?

Question: Should financial data be end-to-end encrypted where even we can't see it?

Current Security:

- HTTPS/TLS for transmission
- Database encryption at rest (Supabase)
- Row Level Security (RLS) for access control
- We (developers) can technically access user data if needed for support

End-to-End Encryption (E2EE):

- User data encrypted with user's key
- We cannot access data even if we wanted to
- Zero-knowledge architecture
- More private, but harder to provide support/recover accounts

Trade-offs:

- **Pros:** Maximum privacy, appealing to security-conscious users
- **Cons:** Account recovery difficult, customer support harder, more complexity

Recommendation: Not for MVP. Current security (HTTPS + RLS + encryption at rest) is sufficient for 95% of users.

Future: Offer as optional "Privacy Mode" for users who want it (Phase 3+).

Impact: Very High if implemented - Major architectural change

18. Performance: Transaction Limit Per Project?

Question: Should we limit the number of transactions per project for performance reasons?

Considerations:

- Power users might log 10+ transactions daily (300/month, 3,600/year)
- IndexedDB can handle 10,000+ records easily
- PostgreSQL with proper indexing can handle millions
- Frontend rendering might slow with 1,000+ items in list

Solutions:

- Pagination (already planned)
- Virtual scrolling for long lists (already planned)
- Archive old transactions (optional user action)
- Lazy loading

Recommendation: No hard limit. Rely on pagination and virtual scrolling. If performance issues arise post-launch, implement archiving.

Impact: Low - Already mitigated with technical approach

19. Localization & Multi-Language Support

Question: Should we support multiple languages in MVP?

Current Plan: English only

Considerations:

- Target audience includes Uganda (English official language, but Swahili/Luganda common)
- Multi-language adds complexity: translations, RTL support, date/currency formats
- Can reach wider audience

Options:

- **MVP:** English only
- **Phase 2:** Add 2-3 languages (Swahili, French, Spanish)
- **Phase 3:** Crowdsourced translations for many languages

Recommendation: English only for MVP. Ensure architecture supports i18n for future:

- Use i18n library (react-i18next)

- Externalize strings to translation files
- Don't hard-code text in components

Impact: Medium - Plan for it now, implement later

20. App Naming & Branding

Question: Is "Headroom" the final name? Is "Your Financial Breathing Space" the final tagline?

Current State:

- Name: Headroom
- Tagline: Your Financial Breathing Space

Considerations:

- Name is memorable, conceptually aligned
- Tagline clearly communicates value
- Domain availability: Check if headroom.app / headroom.com available

Action Items:

- Verify domain availability
- Check trademark conflicts
- Consider alternatives if issues found

Recommendation: Proceed with "Headroom" unless legal/availability issues found. Tagline can evolve based on user feedback.

Impact: Low - Mostly branding/marketing concern