Echo: AI Accountability Coach - Product Requirements Document

Version: 1.0
Last Updated: February 16, 2026
Status: Approved for Development
Owner: Product Team
Contributors: Engineering, Design, Data Science

Executive Summary

Echo is a macOS-native productivity application that transforms conversational planning into actionable task lists, runs distraction-aware focus sessions, enforces app and website blocking, and delivers longitudinal analytics with friend-based accountability. The product combines AI-driven task extraction from natural language with real-time session monitoring and social accountability mechanics to create a comprehensive system that turns vague intentions into measurable outcomes.

Core Value Proposition: Echo plans your life for you through conversation, then holds you accountable to that plan through tracked sessions, enforced blocking, and social pressure.

Target Users: Knowledge workers, students, creators, and remote professionals aged 18-35 who struggle with consistent focus, lack long-term productivity visibility, and want structured accountability without manual overhead.

Business Objectives:

- Achieve 50,000 downloads in first 6 months post-launch
- Maintain 40% Week-1 retention (users who complete 3+ sessions)

- Drive 15% conversion to premium tier (advanced blocking, unlimited history, team features)
- Build viral coefficient of 1.3 through friend invites and accountability groups

Problem Statement

User Problems

Primary Problem: Knowledge workers can articulate what they want to accomplish but consistently fail to follow through, with three root causes:

- **Planning overhead** - Converting vague intentions ("work on thesis") into concrete, time-boxed tasks requires cognitive effort users avoid
- **Distraction invisibility** - Users can't see patterns in when, how often, and why they get distracted during work sessions
- **Accountability gaps** - Social commitments with friends are ad-hoc (group chats, random check-ins) and easy to ignore or abandon

Secondary Problems:

- No longitudinal view of productivity trends (week-over-week, month-over-month)
- Existing time trackers require manual start/stop and don't capture distraction context
- App blockers are binary (on/off) without intelligence about work context or session goals
- Productivity data lives in silos, making it hard to correlate behavior with outcomes

Market Context

The productivity app market is saturated with point solutions (time tracking, blocking, task management), but few products integrate planning, execution tracking, enforcement, and social accountability into a cohesive workflow. Competitors like RescueTime focus on passive tracking without enforcement, Freedom focuses on blocking without task context, and Focusmate focuses on social accountability without automatic planning or analytics.

Market Gap: No product combines conversational AI planning with real-time distraction tracking, intelligent blocking, and friend-based accountability in a single native macOS experience.

Goals and Non-Goals

Goals

V1.0 Launch Goals:

1. Deliver conversational task planning that converts morning chats into projects and tasks with <2 minutes of user effort
2. Track focus sessions with automatic distraction detection (app switches, idle time, blocked site attempts) at <1 second latency
3. Enforce app and website blocking on macOS with configurable modes (soft warn, hard block, timed lock)
4. Surface weekly and monthly productivity stats (total focus time, distraction rate, task completion, streaks)
5. Enable friend invites and shared accountability with lightweight social proof (session counts, streaks, encouragement)

Technical Goals:

- Native SwiftUI application with Liquid Glass visual design language
- Offline-first architecture with sync for tasks, sessions, and social data
- <50 MB memory footprint during active session tracking
- <5% CPU usage during background monitoring
- Zero data loss during network outages or app crashes

Business Goals:

- 40% Week-1 retention (3+ sessions completed)
- 25% Monthly Active User rate
- 15% friend invite acceptance rate
- 10% premium conversion within 30 days of install

Non-Goals (V1.0)

Out of Scope:

- Windows or Linux support (macOS only)
- Team management features (admin dashboards, org policies)
- Calendar integration (Google Calendar, Outlook sync)
- Advanced AI coaching (personalized intervention strategies, adaptive blocking)
- Mobile-first experience (iOS companion is supplementary for blocking only)
- Third-party integrations (Slack, Notion, Asana connectors)
- Pomodoro timer or structured time-blocking methodologies
- Gamification beyond basic streaks (no points, levels, badges, leaderboards)

Future Roadmap Candidates (Post-V1):

- iOS standalone app with full feature parity
- Cross-platform support (Windows, Linux, web)
- Team accountability features (manager visibility, org-wide stats)
- Integration marketplace (API-first architecture for third-party tools)
- Advanced AI coach (predictive distraction alerts, adaptive scheduling)

User Personas

Primary Persona: Alex (Graduate Student)

Demographics:

- Age: 24
- Education: PhD candidate in Computer Science
- Location: Urban area (San Francisco, Boston, New York)
- Tech proficiency: High (comfortable with terminal, config files, keyboard shortcuts)

Context:

- Works from home or coffee shops 4-5 days per week

- Juggling thesis research, TA responsibilities, side projects
- Struggles with starting work (activation energy) and staying focused (distractions)
- Uses Notion for notes, Todoist for tasks, Forest for blocking (but doesn't sync behavior)

Pain Points:

- Planning overhead: "I know what I need to do, but turning it into a checklist feels like work"
- Distraction blindness: "I think I'm focused, but hours disappear to Reddit and Twitter"
- Solo accountability: "No one checks if I actually did what I said I'd do this week"

Jobs to be Done:

- "Help me start my day by turning my brain dump into a concrete plan"
- "Show me exactly when and how I get distracted so I can fix it"
- "Give me social pressure so I can't just bail on my commitments"

Secondary Persona: Jordan (Freelance Designer)

Demographics:

- Age: 29
- Education: Bachelor's in Graphic Design
- Location: Remote (mid-sized city)
- Tech proficiency: Medium (comfortable with design tools, less so with dev tools)

Context:

- Multiple clients with overlapping deadlines
- Works in 2-4 hour blocks between meetings
- Gets paid per project, so wasted time = lost income
- Uses Toggl for time tracking (manual) and Cold Turkey for blocking (too rigid)

Pain Points:

- Context switching: "I lose 20 minutes every time I switch between client projects"
- Inconsistent blocking: "I turn off Cold Turkey when it gets annoying, then forget to turn it back on"
- No visibility: "I can't tell if I'm actually getting faster or just busier"

Jobs to be Done:

- "Track focus time per project without manual timers"
- "Block distractions only during scheduled work sessions, not all day"
- "See month-over-month trends so I can quote projects accurately"

User Journeys

Journey 1: Morning Planning Flow

Actor: Alex (graduate student)
Frequency: Daily (weekday mornings)
Duration: 2-5 minutes

Steps:

1. Alex opens Echo (launches from Dock or Spotlight)
2. Chat interface appears with prompt: "What's on your mind today?"
3. Alex types or speaks: "I need to finish the intro section of my thesis, respond to my advisor's email, prep for my TA session this afternoon, and maybe work on that side project if I have time"
4. Echo parses input and generates structured plan:
    - **Project:** Thesis - Chapter 1
        - Task: Write introduction section (est. 2 hours, priority: High)
        - Task: Incorporate advisor feedback (est. 30 min, priority: High)
    - **Project:** Teaching
        - Task: Prep TA session slides (est. 45 min, priority: Medium)

- **Project:** Side Projects
  - Task: Review open GitHub issues (est. 1 hour, priority: Low)

5. Alex reviews plan, adjusts estimates, reorders priorities
6. Echo asks: "Which apps or sites should I block today?"
7. Alex selects from suggestions (Twitter, Reddit, YouTube) or adds custom entries
8. Alex clicks "Start First Session" or schedules sessions for later

Success Criteria:

- 90% of plans require zero manual edits
- <2 minutes from opening Echo to having a concrete task list
- <10 seconds latency for AI parsing and task generation

Journey 2: Focus Session Execution

Actor: Alex (graduate student)
Frequency: 3-5 times per day
Duration: 45 minutes to 2 hours per session

Steps:

1. Alex clicks "Start Session" from main window or menu bar
2. Echo displays session timer, active task list, and distraction counter
3. Alex works on "Write introduction section" task
4. 15 minutes in, Alex switches to Twitter (blocked app)
   - Echo shows blocking overlay: "Twitter is blocked during this session. Override?"
   - Alex clicks "Stay Focused" (returns to work)
   - Distraction event logged: Twitter attempt at 15:22, duration 8 seconds
5. 30 minutes in, Alex goes idle for 6 minutes (bathroom break, coffee)
   - Echo detects idle time, logs as "natural break" (not counted as distraction)
6. 45 minutes in, Alex completes task, checks box in session panel
7. Alex continues to next task ("Incorporate advisor feedback")
8. 1 hour 15 minutes in, Alex's phone buzzes (text message)
   - iOS companion app blocks notification (configured blocker)

        - Alex doesn't see message until session ends
9. 1 hour 30 minutes in, Alex clicks "End Session"
10. Echo shows session summary:
    - Total time: 1h 30m
    - Focused time: 1h 22m
    - Distractions: 2 (Twitter attempt, accidental Slack switch)
    - Tasks completed: 2/2
    - Streak: 5 days
11. Alex sees notification: "Jordan finished a 2-hour session. Send encouragement?"
12. Alex clicks "Nice work!" which sends to Jordan as push notification

Success Criteria:

- <1 second latency for distraction detection and blocking overlay
- <5% false positive rate (flagging legitimate work as distraction)
- 90% of sessions end with user clicking "End Session" (not force-quit or crash)
- Session summary loads in <500ms

Journey 3: Weekly Review and Social Accountability

Actor: Alex (graduate student)
Frequency: Weekly (Sunday evening)
Duration: 5-10 minutes

Steps:

1. Alex opens Echo, clicks "Stats" tab
2. Weekly dashboard displays:

Metric	This Week	Last Week
Total sessions	18	15
Focused time	22h 15m	19h 40m
Avg session length	1h 14m	1h 18m
Distraction rate	8.2%	11.5%
Tasks completed	34	28
Streak	5 days	4 days

Table 3: Weekly productivity comparison

3. Echo insight notification: "You're 13% more focused this week! Top distractor: Slack (6 attempts). Consider muting channels during sessions."
4. Alex clicks "Friends" tab, sees accountability group stats:
    - Jordan: 16 sessions, 20h 5m focused
    - Sam: 12 sessions, 15h 30m focused
    - Alex: 18 sessions, 22h 15m focused (leading this week)
5. Sam's streak broke (0 sessions yesterday)
6. Alex sends "accountability ping": "Sam, you got this! Start one session today ▯"
7. Sam receives push notification, opens Echo, starts session
8. Echo shows monthly trends graph (line chart of weekly focused time over 8 weeks)
9. Alex exports stats as CSV for personal records

Success Criteria:

- Stats dashboard loads in <1 second with 3 months of history
- Insights are actionable (specific app or behavior identified)
- 30% of users send at least one social interaction per week
- Monthly trend graph accurately reflects cumulative behavior

Functional Requirements

FR-1: Conversational Task Planning

Description: Echo must parse natural language input (text or voice) and extract projects, tasks, estimates, priorities, and dependencies.

Acceptance Criteria:

- AC-1.1: User can type or dictate planning input via chat interface
- AC-1.2: AI extracts entities: Project name, Task title, Time estimate, Priority (High/Medium/Low), Due date (if mentioned)
- AC-1.3: Tasks are organized hierarchically under projects
- AC-1.4: User can edit, reorder, delete, or add tasks in generated plan
- AC-1.5: Plan persists locally and syncs to backend within 5 seconds
- AC-1.6: Parsing latency <10 seconds for inputs up to 500 words

- AC-1.7: Accuracy target: 90% of tasks extracted correctly without manual correction

Technical Implementation:

- Frontend: SwiftUI chat interface with text input and speech-to-text (SFSpeechRecognizer)
- AI Layer: Structured extraction via JSON schema (function calling pattern)
- Model: GPT-4 or Claude 3 with few-shot examples tuned for task extraction
- Fallback: If API fails, allow manual task entry with template form

Edge Cases:

- Ambiguous input ("work on the thing") → Prompt for clarification
- Very long input (>1000 words) → Warn user and truncate or chunk
- Offline mode → Queue input, process when connection restored

FR-2: Focus Session Tracking

Description: Echo must track focus sessions with start/stop controls, task checklist binding, automatic distraction detection, and session summary generation.

Acceptance Criteria:

- AC-2.1: User can start, pause, resume, stop session from main window or menu bar
- AC-2.2: Active session displays: elapsed time, current task, distraction count
- AC-2.3: User can check off tasks during session
- AC-2.4: System logs distraction events: app switch to blocked app, website visit to blocked domain, manual "I got distracted" flag
- AC-2.5: Idle time >5 minutes logged as "break" (not distraction) unless user flags otherwise

- AC-2.6: Session end triggers summary view: total time, focused time, distractions, tasks completed, streak update
- AC-2.7: Session data persists and syncs within 10 seconds of session end

Technical Implementation:

- Session daemon: Background process monitors active app (NSWorkspace), window title, idle time (CGEventSource)
- Event model: SessionEvent stream with types (START, PAUSE, RESUME, STOP, TASK_COMPLETE, DISTRACTION, IDLE)
- Storage: Local SQLite with event sourcing pattern, materialized views for aggregates
- Sync: Append-only log uploaded to backend, conflict-free replication (CRDT or last-write-wins with vector clock)

Performance Requirements:

- Distraction detection latency: <1 second from app switch to event log
- Memory footprint: <50 MB during active session
- CPU usage: <5% during monitoring
- Battery impact: <2% per hour of tracking

Edge Cases:

- Force quit during session → Auto-save state, resume option on relaunch
- System sleep during session → Pause timer, resume when system wakes
- Clock changes (timezone, DST) → Use monotonic clock for elapsed time

FR-3: App and Website Blocking

Description: Echo must block specified apps and websites during focus sessions with configurable modes (soft warn, hard block, timed lock) and override mechanisms.

Acceptance Criteria:

- AC-3.1: User can configure blocklist: apps (by bundle ID) and websites (by domain pattern)
- AC-3.2: Blocking modes:
  - Soft warn: Show overlay, log distraction, allow continue
  - Hard block: Prevent launch or navigation, require override
  - Timed lock: Hard block with 10-second cooldown before override option appears
- AC-3.3: Override flow requires friction: button hold (3 seconds), reason entry, or friend approval (optional)
- AC-3.4: Blocking activates automatically when session starts, deactivates when session ends
- AC-3.5: User can toggle blocking on/off for specific session (e.g., "I need Slack for this task")
- AC-3.6: iOS companion app blocks configured apps and notifications during synced sessions

Technical Implementation:

- macOS app blocking: Launch agent watches for app launch (NSWorkspace), terminates process or hides window
- macOS website blocking: Safari extension (content blocker) or NetworkExtension (DNS/IP filtering)
- iOS blocking: Screen Time API (FamilyControls framework) for app blocking, notification suppression via Focus mode integration
- Blocklist storage: Local plist or SQLite, synced to backend
- Override log: Store reason, timestamp, session context for analytics

Security and Privacy:

- No keylogging or content inspection
- Website blocking via domain patterns only (no URL path inspection)
- User can exempt apps (e.g., password manager, system utilities)

Edge Cases:

- Blocked app launched via Terminal or Automator → Detect and block by process name

- Website blocking in non-Safari browser (Chrome, Firefox) → Require NetworkExtension or warn user
- iOS blocking disabled by user in Settings → Detect and notify in Echo desktop app

FR-4: Productivity Analytics

Description: Echo must aggregate session data into daily, weekly, and monthly views with key metrics, trends, and actionable insights.

Acceptance Criteria:

- AC-4.1: Dashboard displays metrics:
  - Total focused time (sum of session durations minus distractions)
  - Number of sessions completed
  - Average session length
  - Distraction rate (distraction time / total time)
  - Tasks completed vs planned
  - Current streak (consecutive days with 1+ session)
- AC-4.2: Time range selectors: Today, This Week, This Month, Last 7 Days, Last 30 Days, Custom Range
- AC-4.3: Comparison view: side-by-side metrics for selected period vs previous period (e.g., this week vs last week)
- AC-4.4: Trend graphs: line chart of daily or weekly focused time over selected range
- AC-4.5: Top distractors list: ranked by frequency and time lost
- AC-4.6: Insights engine surfaces notifications:
  - Positive reinforcement: "You're 15% more focused this week!"
  - Regression alerts: "Your streak broke. Start one session today to get back on track."
  - Behavior patterns: "You're most productive 9-11 AM. Schedule hard tasks then."
  - Distractor callouts: "Twitter cost you 45 minutes this week. Consider blocking it."
- AC-4.7: Data export: CSV or JSON format with session-level and aggregate data

Technical Implementation:

- Aggregation: Materialized views in SQLite updated after each session
- Backend: Postgres with time-series optimizations (TimescaleDB or partitioned tables)
- Insights engine: Rules-based system (e.g., "If distraction_rate increases >20% week-over-week, trigger alert")
- Graphs: SwiftUI Charts framework with interactive tooltips
- Export: Background task generates CSV, stores temporarily, allows share via macOS share sheet

Performance Requirements:

- Dashboard load time: <1 second for 3 months of data
- Graph rendering: <500ms for 90 days of data points
- Insight generation: Background job runs hourly, latency not user-visible

Edge Cases:

- No sessions in selected period → Show empty state with encouragement
- Incomplete session (app crashed) → Mark as partial, include in stats with note
- Data sync conflict (desktop and mobile sessions overlapping) → Use vector clock or server timestamp for resolution

FR-5: Social Accountability

Description: Echo must enable friend invites, accountability groups, shared stats, and lightweight social interactions (encouragement, pings, reactions).

Acceptance Criteria:

- AC-5.1: User can invite friends via shareable link or email
- AC-5.2: Invited friend receives notification, clicks link, installs app (or opens if installed), accepts friend request
- AC-5.3: Friends see each other's high-level stats: session count, total focused time, current streak (no task details or distraction logs unless explicitly shared)

- AC-5.4: User can create accountability group (2-10 people) with shared leaderboard
- AC-5.5: Leaderboard displays weekly rankings by focused time or session count
- AC-5.6: Social interactions:
  - Encouragement: "Nice work!" or custom message sent as push notification
  - Accountability ping: "Start a session!" with optional message
  - Reactions: thumbs up, fire emoji on completed sessions
- AC-5.7: Privacy controls: user can set visibility (all friends, specific group, private)
- AC-5.8: Notifications: push (iOS, macOS) and in-app for social events

Technical Implementation:

- Backend: Social graph stored in Postgres (users, friendships, groups, membership)
- Invite system: Generate unique invite code, track referrer for attribution
- Privacy: Row-level security in DB, visibility flag on user profile
- Notifications: APNs for iOS, local notifications for macOS, in-app notification center
- Real-time updates: WebSocket or long polling for leaderboard updates (optional, not V1 critical)

Social Mechanics Design Principles:

- Lightweight: No mandatory social interaction, opt-in for users who want accountability
- Positive: Focus on encouragement, not shame or competition
- Respect privacy: Default to private, explicit opt-in for sharing

Edge Cases:

- Friend request declined → Remove pending invite, allow re-invite after 7 days
- User leaves group → Remove from leaderboard, notify group members

- Spam prevention → Limit invites to 20 per day, rate-limit pings to 5 per friend per day

FR-6: iOS Companion App

Description: Echo iOS companion app provides blocking enforcement and lightweight session visibility, syncing with macOS primary app.

Acceptance Criteria:

- AC-6.1: iOS app syncs blocklist from macOS app
- AC-6.2: When macOS session starts, iOS app activates blocking (apps and notifications)
- AC-6.3: iOS app displays active session timer and current task (read-only)
- AC-6.4: User can end session from iOS, which syncs to macOS
- AC-6.5: iOS app shows daily stats summary (focused time, session count, streak)
- AC-6.6: Push notifications for social events and insights

Technical Implementation:

- iOS app: SwiftUI, Screen Time API for blocking, push notifications via APNs
- Sync: Shared backend API, periodic polling (every 30 seconds) or push-based updates
- Session control: iOS can end session, but cannot start or create tasks (macOS-only)

V1 Scope Limitation:

- iOS is supplementary, not standalone
- No task creation, planning, or editing on iOS
- No analytics dashboard on iOS (summary stats only)

Non-Functional Requirements

NFR-1: Performance

- Page load times: <1 second for all views with 3 months of data
- Distraction detection latency: <1 second from trigger event to UI update
- Blocking enforcement latency: <500ms from app launch to block overlay
- Memory footprint: <100 MB total (daemon + UI), <50 MB during active session
- CPU usage: <5% during monitoring, <10% during AI parsing
- Battery impact: <3% per hour of active tracking on MacBook
- Sync latency: <10 seconds from local write to backend acknowledgment

NFR-2: Reliability

- Uptime target: 99.5% for backend API (downtime allowance: 3.6 hours/month)
- Data durability: Zero session data loss during app crash, network failure, or system reboot
- Crash rate: <0.1% of sessions (1 crash per 1000 sessions)
- Sync conflict resolution: Deterministic, user-understandable (last-write-wins with timestamp display)

NFR-3: Security and Privacy

- Data encryption: AES-256 at rest (local DB), TLS 1.3 in transit (API)
- Authentication: OAuth 2.0 with refresh tokens, optional biometric unlock (Touch ID, Face ID)
- No third-party analytics or tracking SDKs in V1 (minimize privacy surface area)
- User data deletion: Account deletion triggers immediate local wipe and backend cascade delete within 24 hours
- Compliance: GDPR-compliant data handling, privacy policy published before launch

NFR-4: Accessibility

- VoiceOver support: All UI elements labeled, navigable via keyboard and screen reader
- Keyboard shortcuts: Power users can control all primary functions without mouse
- Reduced motion: Disable Liquid Glass animations if system preference set
- Reduced transparency: Fall back to opaque backgrounds if system preference set
- Color contrast: WCAG AA compliance (4.5:1 for normal text, 3:1 for large text)

NFR-5: Localization (Future)

- V1: English (US) only
- V2 target: English (UK), Spanish, French, German, Japanese
- String externalization: All UI text in localizable strings files from day one

Technical Architecture

System Overview

Echo follows a client-server architecture with macOS and iOS clients syncing to a shared backend. The macOS app is the primary interface for planning, session management, and analytics, while the iOS app provides blocking enforcement and lightweight visibility.

Architecture Diagram (Conceptual):

[Client Layer]
\newline
macOS App (SwiftUI + AppKit) ↔ Session Daemon (background process)
\newline
iOS App (SwiftUI + Screen Time API)
\newline
↓ HTTPS/TLS 1.3 ↓
\newline
[Backend Layer]

\newline
API Server (Fastify/TypeScript) ↔ Postgres DB ↔ Redis (cache)
\newline
↓
\newline
AI Service (GPT-4 API) | Push Notification Service (APNs)
Figure 1: Echo system architecture

macOS Client

Tech Stack:

- **UI Framework:** SwiftUI (primary), AppKit (menu bar, system integration)
- **State Management:** Swift Concurrency (async/await, actors)
- **Local Storage:** SQLite via GRDB.swift (session events, tasks, cache)
- **Networking:** URLSession with Codable for REST API
- **Real-time Monitoring:** NSWorkspace for app tracking, CGEventSource for idle time
- **Blocking:** Launch agent (LSUIElement) for app blocking, NetworkExtension for web filtering (optional)

App Structure:

- **Main App:** User-facing SwiftUI windows (planning, session, stats, social)
- **Session Daemon:** Background agent runs continuously, monitors system events, enforces blocking
- **Menu Bar Agent:** Always-visible menu bar icon with quick actions (start session, view stats)
- **Safari Extension:** Content blocker for website filtering (packaged with main app)

Data Flow (Session Tracking):

1. User clicks "Start Session" in UI
2. UI sends command to session daemon via XPC or distributed notifications
3. Daemon starts monitoring loop: poll active app every 1 second, track idle time via CGEventSource

4. On distraction event (blocked app launched), daemon:
    - Logs event to local SQLite (timestamp, app name, duration)
    - Shows blocking overlay via window management
    - Sends notification to UI (distraction counter increments)
5. User ends session via UI
6. UI sends stop command to daemon
7. Daemon finalizes session record, triggers sync to backend
8. Backend responds with updated stats, UI refreshes dashboard

iOS Client

Tech Stack:

- **UI Framework:** SwiftUI
- **Blocking:** Screen Time API (FamilyControls, ManagedSettings)
- **Notifications:** APNs (remote), UserNotifications (local)
- **Networking:** URLSession with Codable

Scope:

- Blocking enforcement only (no planning or analytics UI)
- Lightweight session visibility (timer, current task read-only)
- Push notifications for social events and insights

Blocking Implementation:

1. macOS app starts session, sends blocklist to backend
2. Backend pushes notification to iOS app via APNs
3. iOS app activates Screen Time restriction (FamilyControls API)
4. User attempts to open blocked app → iOS system shows restriction screen
5. macOS app ends session → Backend notifies iOS → iOS deactivates restriction

Backend

Tech Stack:

- **API Server:** Fastify (TypeScript) or FastAPI (Python)
- **Database:** Postgres 15+ (JSONB for flexible schemas, partitioning for time-series data)
- **Cache:** Redis (session state, leaderboard rankings)

- **Message Queue:** Redis Pub/Sub or AWS SQS (async tasks like insight generation)
- **File Storage:** S3 (exported CSVs, user avatars)
- **Hosting:** AWS (EC2, RDS, ElastiCache) or Vercel/Railway (serverless option)

API Endpoints (Key Routes):

Endpoint	Method	Description
/auth/login	POST	OAuth login, returns JWT
/auth/refresh	POST	Refresh access token
/tasks	GET	Fetch all tasks for user
/tasks	POST	Create task or project
/tasks/:id	PATCH	Update task (status, estimate)
/sessions	GET	Fetch session history
/sessions	POST	Create session record
/sessions/:id/events	POST	Append session event (distraction, task complete)
/sessions/:id	PATCH	End session, finalize duration
/stats	GET	Aggregate stats (daily, weekly, monthly)
/friends	GET	Fetch friend list
/friends/invite	POST	Generate invite link
/friends/:id/accept	POST	Accept friend request
/groups	GET	Fetch accountability groups
/groups/:id/leaderboard	GET	Group leaderboard rankings
/social/encourage	POST	Send encouragement to friend
/social/ping	POST	Send accountability ping

Table 4: Core API endpoints

Database Schema (Simplified):

Table	Columns
users	id, email, auth_provider, created_at, settings_json
projects	id, user_id, name, created_at
tasks	id, project_id, title, estimate_minutes, priority, status, due_date
sessions	id, user_id, start_time, end_time, duration_seconds, focused_seconds, distraction_count
session_events	id, session_id, event_type, timestamp, app_name, duration_seconds, metadata_json
friendships	user_id_1, user_id_2, status (pending/accepted), created_at
groups	id, name, created_by, created_at
group_members	group_id, user_id, joined_at
social_events	id, from_user_id, to_user_id, event_type (encourage/ping), timestamp, message

Table 5: Core database tables

Sync Strategy:

- Event sourcing for sessions: append-only log of session events, materialized views for aggregates
- CRDT or last-write-wins for tasks (with vector clock timestamp)
- Periodic full sync every 24 hours to reconcile any drift

AI Service

Planning AI:

- **Model:** GPT-4 or Claude 3 (Sonnet for cost, Opus for accuracy)
- **Prompt:** Structured extraction with JSON schema for projects, tasks, estimates

- **Few-shot examples:** 5-10 examples of input → output task lists
- **Fallback:** If API times out or returns invalid JSON, prompt user to enter tasks manually

Insights Engine:

- **V1:** Rules-based system (no ML model)
- **Rules:** Hardcoded thresholds (e.g., "If distraction rate >15%, suggest blocking")
- **V2+:** Train classifier on user behavior patterns to predict optimal intervention timing

Liquid Glass UI Design System

Visual Language:

- **Materials:** Translucent backgrounds with blur (NSVisualEffectView on macOS)
- **Depth:** Subtle drop shadows, specular highlights on interactive elements
- **Motion:** Smooth transitions (spring animations), morphing shapes for state changes
- **Typography:** SF Pro (system font), large titles, clear hierarchy
- **Color:** Adaptive (light/dark mode), accent color user-configurable

Implementation Details:

- Use SwiftUI .background(.ultraThinMaterial) for translucent panels
- Custom shape modifiers for rounded, concentric geometry (RoundedRectangle with large corner radii)
- Animation curves: .spring(response: 0.4, dampingFraction: 0.7) for smooth motion
- Accessibility: Fall back to opaque backgrounds if "Reduce Transparency" enabled

Data and Privacy

Data Collection

What Echo Collects:

- Account data: email, authentication tokens, user settings
- Planning data: projects, tasks, estimates, priorities, due dates
- Session data: start time, end time, duration, distraction events (app name, timestamp, duration)
- Social data: friend relationships, group memberships, social interactions (encouragement, pings)
- Analytics: aggregate stats (focused time, session counts, streaks)

What Echo Does NOT Collect:

- Keystroke logs or screen content
- Full URLs or website content (only domain-level blocking)
- Task content (e.g., "write thesis intro" stored, but not thesis text itself)
- Location data
- Microphone or camera access (speech-to-text happens on-device via SFSpeechRecognizer)

Data Storage and Retention

- **Local (macOS):** SQLite database stored in user's Application Support directory, encrypted with FileVault (if enabled by user)
- **Backend:** Postgres database with AES-256 encryption at rest, TLS 1.3 in transit
- **Retention:** Session data retained indefinitely unless user deletes account or manually deletes sessions
- **Backups:** Daily encrypted backups of backend DB, 30-day retention

User Rights (GDPR Compliance)

- **Access:** User can export all data (tasks, sessions, social graph) as JSON or CSV
- **Deletion:** Account deletion triggers immediate local DB wipe and backend cascade delete within 24 hours

- **Portability:** Exported data in machine-readable format (JSON/CSV)
- **Correction:** User can edit or delete any task, session, or social interaction via UI

Third-Party Services

- **OpenAI (GPT-4):** Task planning input sent to API, no storage by OpenAI per zero-retention agreement
- **Apple (APNs):** Push notification tokens and payloads, no user content stored by Apple
- **AWS:** Backend hosting, data encrypted and access-controlled per AWS best practices

Testing and Quality Assurance

Testing Strategy

Unit Tests:

- Target: 80% code coverage for business logic (task parsing, session aggregation, blocking logic)
- Framework: XCTest (Swift), Jest (TypeScript backend)
- Key modules: AI parsing validation, session event aggregation, distraction detection logic

Integration Tests:

- API contract tests: Validate request/response schemas for all endpoints
- Sync tests: Ensure local changes propagate to backend and vice versa
- Blocking tests: Verify app and website blocking works across macOS versions

End-to-End Tests:

- User journey automation: Planning flow, session execution, weekly review
- Framework: XCUITest (macOS), Fastlane Snapshot for screenshots

- Target: 100% coverage of critical paths (FR-1 through FR-5)

Performance Tests:

- Load testing: Backend API under 1000 concurrent users
- Stress testing: macOS app with 10,000 session records in local DB
- Battery testing: Measure battery drain over 8-hour workday with active tracking

Quality Metrics

Metric	Target	Measurement
Crash-free rate	>99.9%	Crashlytics or Sentry
Unit test coverage	>80%	Xcode coverage report
API uptime	>99.5%	Pingdom or UptimeRobot
User-reported bugs	<5 per 1000 users	Support ticket system
Session data loss	0%	Audit logs + user reports
Distraction detection accuracy	>95%	User feedback + manual QA

Table 6: Quality assurance targets

Testing Plan (Pre-Launch)

Alpha Testing (Internal):

- Duration: 2 weeks
- Audience: Engineering team + design team (10-15 people)
- Focus: Core functionality, blocking enforcement, crash stability

Beta Testing (External):

- Duration: 4 weeks
- Audience: 200-500 early adopters recruited via ProductHunt, Twitter, Reddit
- Focus: Real-world usage patterns, edge cases, feedback on UX and AI parsing accuracy

- Instrumentation: Opt-in anonymous usage analytics (session counts, crashes, feature adoption)

Release Candidate:

- Duration: 1 week
- Audience: 1000+ users from beta waitlist
- Focus: Final bug fixes, performance validation, backend load testing

Launch and Go-to-Market

Launch Timeline

Milestone	Date	Deliverables
Kickoff	Week 0	PRD finalized, tech stack selected, team assigned
Alpha Build	Week 8	Core features functional, internal testing begins
Beta Launch	Week 12	Public beta with 200+ users, feedback loop
RC Build	Week 16	All P0 bugs fixed, backend scaled for launch
Public Launch	Week 18	v1.0 on website, ProductHunt, social media
Post-Launch	Week 20-24	Monitor metrics, iterate on feedback, plan v1.1

Table 7: Development and launch timeline

Go-to-Market Strategy

Target Channels:

1. **ProductHunt:** Launch post with demo video, founder story, limited-time early access
2. **Twitter/X:** Founder-led content (productivity tips, AI behind-the-scenes), influencer partnerships

3. **Reddit:** r/productivity, r/getdisciplined, r/macapps (organic posts, AMAs)
4. **YouTube:** Demo video, "day in the life" with Echo, productivity creator sponsorships
5. **Email:** Beta waitlist (5000+ signups), launch announcement, drip campaign

Pricing Strategy (V1):

- **Free Tier:** Unlimited sessions, 30 days of history, 3 friends, basic blocking
- **Pro Tier ($8/month or $60/year):** Unlimited history, unlimited friends, advanced blocking (timed locks, friend approval), priority support
- **Team Tier ($15/user/month):** Org dashboards, admin controls, team accountability groups (post-V1)

Launch Goals:

- 10,000 downloads in first week
- 500 ProductHunt upvotes (top 5 product of the day)
- 40% Week-1 retention (users who complete 3+ sessions)
- 10% free-to-paid conversion within 30 days

Marketing Messaging

Value Propositions:

1. **Zero Planning Overhead:** "Just talk to Echo in the morning. Get a concrete plan in 2 minutes."
2. **See Your Distractions:** "Stop wondering where your time goes. Echo shows you exactly when and why you lose focus."
3. **Social Accountability:** "Your friends see your progress. You can't fake it."
4. **Liquid Glass Design:** "The most beautiful productivity app on Mac. Period."

Target Messaging by Persona:

- **Alex (grad student):** "Thesis not writing itself? Echo keeps you accountable."

- **Jordan (freelancer):** "Stop losing money to distractions. Track every focused minute."

Success Metrics and KPIs

North Star Metric

Weekly Focused Time per Active User: Total focused minutes (session duration minus distractions) per weekly active user. Target: 15+ hours/week (representing ~3 hours/day for 5 days).

Key Performance Indicators

Metric	Target (30 days post-launch)	Measurement
Downloads	50,000	Website analytics + DMG installs
Activation rate	60% (first session completed)	Backend telemetry
Week-1 retention	40% (3+ sessions in first 7 days)	Cohort analysis
Week-4 retention	25%	Cohort analysis
Daily Active Users	10,000	Backend telemetry
Weekly Active Users	25,000	Backend telemetry
Avg sessions/user/week	10	Session logs
Avg focused time/user/week	15 hours	Session aggregates
Friend invite rate	15% (users who invite 1+ friend)	Invite API logs
Invite acceptance rate	20%	Invite API logs
Free-to-paid conversion	10% within 30 days	Subscription API
Churn rate	<5% monthly (paid users)	Subscription cancellations
NPS score	>50	In-app survey

Table 8: Success metrics and targets

Analytics Instrumentation

Events to Track:

- App launched, first session started, first task completed
- Session started, session ended, distraction detected, blocking triggered, override requested
- Task created, task completed, project created
- Friend invited, friend accepted, social interaction sent
- Stats dashboard viewed, export triggered
- Premium upgrade started, premium upgrade completed

Privacy-First Analytics:

- All events anonymized (user ID hashed)
- No PII in event metadata (no task titles, app names are categorical)
- Opt-out available in settings
- No third-party analytics SDKs (self-hosted or privacy-focused alternatives like PostHog)

Risks and Mitigations

Technical Risks

Risk	**Likelihood**	**Mitigation**
App blocking fails on macOS updates	Medium	Test on beta versions, fallback to soft warnings
AI parsing accuracy <80%	Medium	Few-shot tuning, user feedback loop, manual override
Backend cannot scale to 10K concurrent users	Low	Load testing, horizontal scaling plan, CDN for static assets
iOS Screen Time API restrictions	High	Document limitations, manage user expectations
Data sync conflicts	Medium	CRDT or vector clock, deterministic resolution, user notification
Battery drain >5%/hour	Low	Profile and optimize, background throttling

Table 9: Technical risk register

Business Risks

Risk	**Likelihood**	**Mitigation**
Low retention (<20% Week-1)	Medium	Onboarding improvements, push notification engagement
Users turn off blocking	High	Gamify blocking (streaks), friend approval required for override
Social features unused (<10% adoption)	Medium	Prompt users to invite friends during onboarding
Premium conversion <5%	Medium	Time-limited trials, upsell prompts in stats view
Competitor launches similar product	Low	Speed to market, focus on Liquid Glass design differentiation

Table 10: Business risk register

Privacy and Legal Risks

Risk	Likelihood	Mitigation
GDPR non-compliance	Low	Legal review, data export/deletion features, privacy policy
User perceives app as spyware	Medium	Transparency: no keystroke logging, no content inspection, open-source monitoring code
App Store rejection (blocking features)	Medium	Submit early for review, provide App Review notes explaining use case

Table 11: Privacy and legal risk register

Future Roadmap (Post-V1)

V1.1 (3 months post-launch)

- Pomodoro mode (25-minute work, 5-minute break cycles)
- Task templates (recurring daily/weekly tasks)
- Improved AI parsing (multi-turn conversation, ask clarifying questions)
- Weekly email digest (stats summary, insights, friend activity)

V1.5 (6 months post-launch)

- Calendar integration (Google Calendar, Outlook sync for scheduling sessions)
- Advanced insights (predictive distraction alerts, optimal scheduling recommendations)
- iOS standalone mode (full feature parity: planning, analytics on iPhone)
- Custom focus modes (deep work, meetings, admin tasks with different blocklists)

V2.0 (12 months post-launch)

- Windows and Linux support (cross-platform Electron or native rewrites)
- Team features (org dashboards, manager visibility, team accountability)
- Integration marketplace (Slack, Notion, Asana, GitHub via API)
- AI coach v2 (adaptive blocking strategies, personalized intervention timing)
- Gamification (points, levels, badges, team challenges)

Appendix

Glossary

- **Accountability group:** A collection of 2-10 friends who share productivity stats and hold each other accountable
- **Blocklist:** User-configured list of apps and websites to block during focus sessions
- **Distraction event:** A logged moment when user attempts to access a blocked app or website, or manually flags distraction
- **Focused time:** Total session duration minus time lost to distractions and breaks
- **Hard block:** Blocking mode that prevents app launch or website access, requires override to continue
- **Liquid Glass UI:** Apple's design language featuring translucent materials, layered blur, smooth animations, and rounded geometry
- **Session:** A timed work period with start and end, during which tasks are tracked and distractions are logged
- **Soft warn:** Blocking mode that shows a warning overlay but allows user to continue without friction
- **Streak:** Number of consecutive days with at least one completed focus session
- **Timed lock:** Blocking mode with a mandatory cooldown period (e.g., 10 seconds) before override option appears

References

[1] Apple. (2025, June 8). Meet Liquid Glass - WWDC25. Apple Developer. https://developer.apple.com/videos/play/wwdc2025/219/

[2] Apple. (2025). Liquid Glass | Apple Developer Documentation. https://developer.apple.com/documentation/TechnologyOverviews/liquid-glass

[3] Echo Productivity. (2026, January 13). Echo - AI Accountability That Plans Everything. https://echoproductivity.com

[4] Echo Productivity. (2026, January 19). Echo for Mac: The Official Launch [Video]. YouTube. https://www.youtube.com/watch?v=NPT1jmTXOl8

Document History

Version	Date	Changes
1.0	Feb 16, 2026	Initial PRD for v1.0 launch

Table 12: Document revision history