FlowBuddy: System Requirements Specification (SRS) v2.5 - FINAL

Version: 2.5 - Agentic Coach with Psychological Safety
Date: January 3, 2026
Project: Comet Resolution v2 Hackathon - Productivity Category
Status: Implementation-Ready

🎯 Product Philosophy: "Supportive Autonomy"

The Reconciliation: FlowBuddy is an **agentic AI coach** (proactive, autonomous, outcome-focused) that operates with **psychological safety principles** (non-judgmental, permission-based, respects cognitive load).

Core Tension Resolved:

- ✅ **Agentic:** Takes initiative, plans autonomously, intervenes predictively
- ✅ **Safe:** Never commands, always offers choices, respects "no"
- ✅ **Transparent:** Shows reasoning, allows undo, explains every action

Voice Example:

- ❌ Pushy Coach: "You missed your workout. Get it done by 6 PM."
- ❌ Passive Tool: "Would you like me to suggest a workout time?"
- ✅ **FlowBuddy:** "I noticed you missed 9 AM. I moved it to 6 PM (your usual backup time). Work for you? [Yes] [Pick different time] [Skip today]"

Table of Contents

---

1. Hackathon Alignment & MVP Scope

1.1 Hackathon Requirements

Comet Resolution v2: Build an AI agent that meaningfully helps people keep their resolutions with measurable impact and observability.

FlowBuddy's Fit:

- ✅ **Agentic Behavior:** 3 autonomous agents (decomposer, scheduler, accountability)
- ✅ **Measurable Impact:** Resolution completion rate, habit consistency, intervention effectiveness
- ✅ **Observability:** Full Opik integration with 12+ metrics (AI quality + user outcomes + guardrails)
- ✅ **Innovation:** Dual-input model (structured resolutions + brain dump signals)

1.2 MVP Feature Prioritization (Realistic 4-Week Build)

P0 (Must Ship - Week 1-3):

Feature	Agentic Level	Build Effort	Hackathon Value
1. Dual Intake: Brain Dump + Resolution Setup	Medium	1 week	Shows UX innovation
2. Resolution Decomposer Agent	High	1 week	Core agentic demo
3. Weekly Micro-Resolution Generator	High	3 days	Autonomous planning
4. Basic Intervention System	Medium	1 week	Outcome driver

P1 (Nice to Have - Week 4):

Feature	Build Effort	If Time Allows
Daily schedule optimization	1 week	Start with manual confirm, add auto later
Pattern recognition	1 week	Start with heuristics (e.g., "90% success on Mon 9 AM")

P2 (Post-Hackathon):

- ML-based pattern prediction
- Advanced background job scheduling
- Deep Work Shield

Strategic Trade-off: We're **cutting daily auto-scheduling** from MVP to focus on:

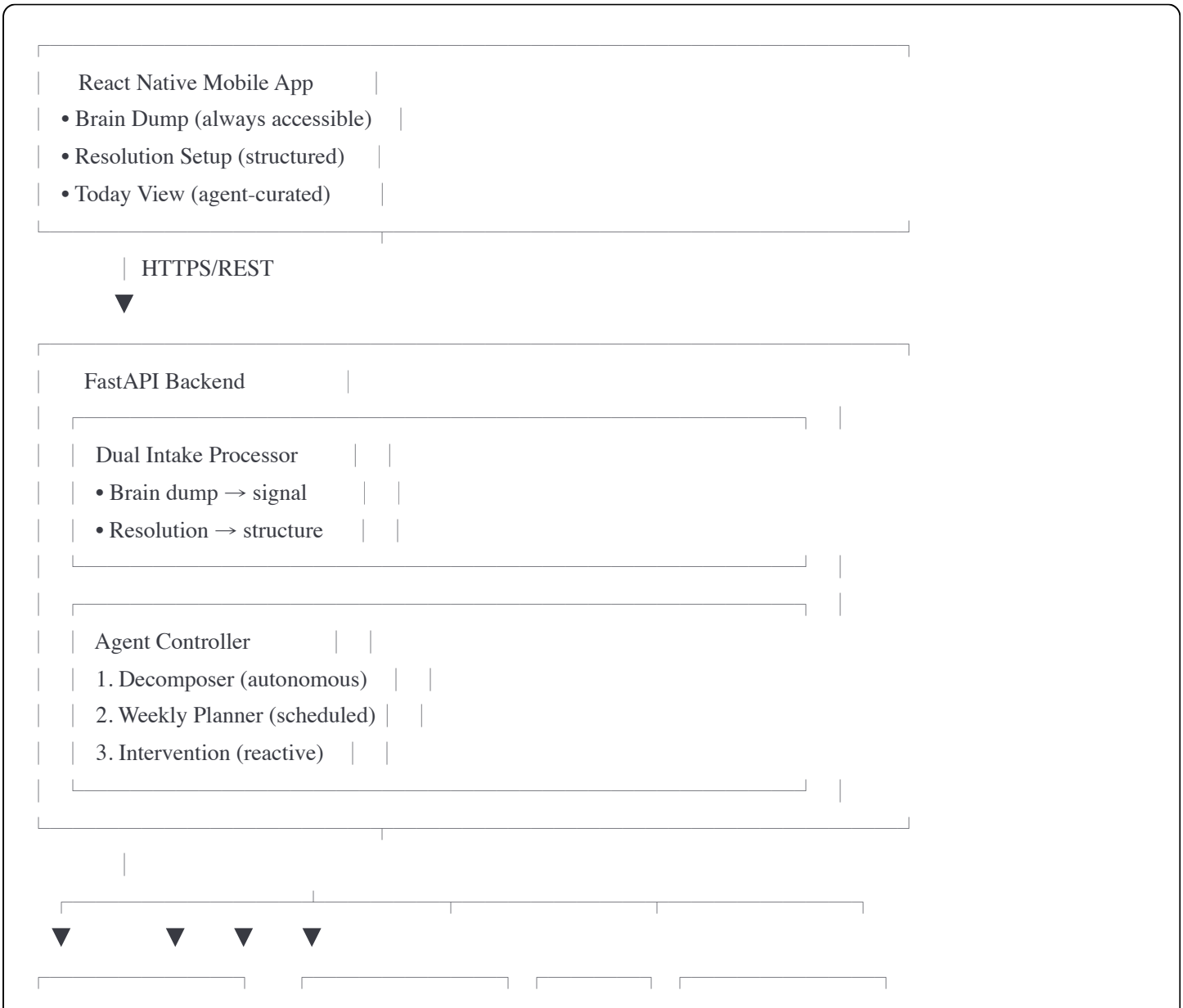
1. Perfect execution of resolution decomposition
2. Robust intervention system
3. Rich Opik observability

This reduces implementation risk while keeping the agentic thesis strong.

---

2. System Overview

2.1 High-Level Architecture



2.2 Agent Responsibilities (Simplified for MVP)

Agent 1: Resolution Decomposer

- **Trigger:** User creates new resolution
- **Autonomy:** Generates 4-12 week plan + Week 1 tasks automatically
- **Permission Model:** User must confirm before activation
- **Opik Traces:** `resolution_decomposition`

Agent 2: Weekly Micro-Resolution Planner

- **Trigger:** Every Sunday 8 PM (automated job)
- **Autonomy:** Generates next week's tasks based on last week's performance
- **Permission Model:** User receives push notification to review/accept
- **Opik Traces:** `weekly_planning`

Agent 3: Intervention Coach

- **Trigger:** Slippage detection (Thursday check if <50% toward goal)
- **Autonomy:** Detects risk, generates personalized recovery plan
- **Permission Model:** Offers 3 options (get back on track / adjust goal / take break)
- **Opik Traces:** `slippage_intervention`

NOT in MVP (but documented for v2):

- Daily auto-scheduling (too complex for 4 weeks)
- ML-based pattern prediction (start with simple heuristics)
- Proactive morning check-ins (save for post-demo)

---

3. User Personas & Journeys

3.1 Primary Persona: "Resolution Rachel"

Demographics:

- Age: 28-35, knowledge worker
- Resolutions: "Exercise 3x/week", "Read 12 books/year", "Launch side project"
- Pain: Starts strong, loses momentum by Week 3, needs accountability without guilt

What She Needs:

1. **Easy start:** Shouldn't take >2 min to set up resolution
2. **Autonomy:** Agent breaks down goal, she confirms
3. **Flexibility:** Can reschedule/adjust without feeling like she "failed"
4. **Clarity:** Always knows if she's on track
5. **Safety valve:** Can "brain dump" overwhelm without it becoming more tasks

3.2 Dual-Input Journey (The Innovation)

Path A: Structured Resolution Setup

User Action: Taps "New Resolution"
Agent: "What resolution would you like to work on?"
User: Types "Exercise 3x per week"

Agent (Autonomous):
- Generates 12-week plan
- Creates Week 1 tasks (Mon 9AM, Wed 7PM)
- Shows for approval

User: Reviews → Taps "Looks good!"
Agent: Locks in, sets Sunday reminder for Week 2 planning

Emotional Arc: Motivated → Supported → Confident

Path B: Brain Dump as Signal (The Differentiator)

```
User Action: Taps "Brain Dump" (always visible button)
User: "Feeling overwhelmed, missed my workout today,
      worried I'm falling behind on exercise goal"

Agent (Signal Processing):
- Detects: emotion (overwhelm), event (missed workout),
  resolution reference (exercise)
- Does NOT create new task (avoids task creep)
- Uses as input signal

Agent Response:
"Got it. Sounds like today was tough. I see you're
at 1/2 workouts this week. Want me to reschedule
Wednesday's session for this weekend instead?"

[Yes] [No, I'll handle it]

If Yes: Agent reschedules autonomously
If No: Agent acknowledges, doesn't push

Emotional Arc: Stressed → Heard → Relieved
```

Key Insight: Brain dump is **NOT** for task creation. It's a **pressure release valve** that gives the agent context about:

- User's emotional state (overwhelmed, motivated, etc.)
- Blockers ("had a meeting conflict")
- Intent shifts ("maybe 3x/week is too much")

This allows the agent to **adapt its interventions** based on qualitative signals, not just completion data.

---

4. MVP Functional Requirements

4.1 Feature 1: Dual Intake System

FR-1.1: Brain Dump (Pressure Relief)

Priority: P0 (Core UX differentiator)
Agentic Level: Medium (Processes for signals, doesn't auto-create tasks)

User Story:
As Rachel, when I'm stressed or have scattered thoughts, I want to dump them quickly so I feel heard, and the agent can use that context to help me without creating more work.

Acceptance Criteria:

- [ ] "Brain Dump" button is always visible (home screen sticky button)
- [ ] Tapping opens fullscreen text input (auto-focused keyboard)
- [ ] User types/pastes unstructured text (max 2000 chars)
- [ ] Submit triggers:
  - **Signal extraction** (LLM identifies: emotions, blockers, resolution references)
  - **Acknowledgment** (empathetic confirmation)
  - **Optional action** (if agent detects actionable intent, offers to help)
- [ ] Does NOT auto-create tasks (avoids overwhelm)
- [ ] All signal processing logged to Opik

Technical Specification:

python

```
# LLM Prompt: Brain Dump Signal Extractor
BRAIN_DUMP_SIGNAL_PROMPT = """
User's brain dump: "{input_text}"
Active resolutions: {user_resolutions}

Extract signals (do NOT create tasks):
1. Emotional state: overwhelmed | stressed | motivated | neutral
2. Resolution references: which resolutions mentioned?
3. Blockers: what's preventing progress?
4. Intent shifts: does user want to adjust goals?

Return JSON:
{{
  "emotional_state": "overwhelmed",
  "resolution_refs": ["Exercise 3x/week"],
  "blockers": ["Meeting conflict on Wednesday"],
  "intent_shift": "possibly wants to reduce frequency",
  "actionable": true,
  "suggested_response": "Sounds like Wednesday is tough. Want me to move workouts to weekends?"
}}

Principles:
- Acknowledge feelings ("tough day")
- Don't judge ("you should have...")
- Offer help, don't command
"""
```

Opik Trace Structure:

```python
{
  "trace_name": "brain_dump_processing",
  "spans": [
    {"name": "signal_extraction", "llm_tokens": 450},
    {"name": "resolution_matching", "vector_similarity": 0.87},
    {"name": "empathy_generation", "llm_tokens": 120}
  ],
  "metadata": {
    "emotional_state": "overwhelmed",
    "actionable": true,
    "user_accepted_help": true
  }
}
```

Example Flow:

```
User: "Ugh missed my run today, work was crazy"
Agent: "Got it. Sounds like today was packed. You're
      at 1/2 workouts this week. Want me to move
      Wednesday's to Saturday 10 AM instead?"
      [Yes, reschedule] [No thanks]
```

Edge Cases:

- No resolution reference → Just acknowledge: "Thanks for sharing. I've noted this."
- Pure emotional vent → Supportive response: "Rough day. You've got this."
- Actionable but user says "No thanks" → Agent backs off, doesn't push

---

FR-1.2: Resolution Setup (Structured Intake)

Priority: P0 (Core agentic feature)
Agentic Level: High (Autonomous decomposition)

User Story:
As Rachel, I want to tell the agent my big goal and have it break it down into achievable milestones so I don't have to plan everything myself.

Acceptance Criteria:

- [ ] User navigates to "New Resolution"
- [ ] Conversational form: "What resolution would you like to work on?"
- [ ] User types free text (e.g., "Exercise 3x per week")

- [ ] Agent autonomously:
  - Classifies type (habit, project, learning)
  - Generates 4-12 week milestone plan
  - Creates Week 1 specific tasks (2-5 tasks with suggested times)
- [ ] Shows plan for user approval
- [ ] User can: Accept / Edit tasks / Regenerate with different approach
- [ ] Once accepted, agent sets Sunday reminder for Week 2 planning

LLM Prompt:

```python
```

```
RESOLUTION_DECOMPOSER_PROMPT = """
User's resolution: "{resolution_text}"
Today: {current_date}
User context: {user_schedule_hints}

Generate a realistic plan:
1. Classify: habit | project | learning | health
2. Set duration: 4-12 weeks (prefer 12 for habits)
3. Create milestones:
   - Week 1-2: Easier (build momentum)
   - Week 3+: Full target
4. For Week 1, generate 2-4 specific tasks with times

Return JSON:
{{
  "type": "habit",
  "title": "Exercise 3x per week",
  "duration_weeks": 12,
  "milestones": [
    {{
      "week": 1,
      "target": "2 sessions (warmup)",
      "tasks": [
        {{"title": "30-min jog", "day": "Monday", "time": "09:00", "duration_min": 30}},
        {{"title": "Yoga class", "day": "Wednesday", "time": "19:00", "duration_min": 45}}
      ]
    }}
  ]
}}

Coaching principles:
- Week 1 should feel achievable (not overwhelming)
- Tasks must be specific (activity + duration)
- Respect common work schedules (avoid midday)
"""
```

Opik Metrics:

- `decomposition_quality` (LLM-as-Judge: Are tasks specific and achievable?)
- `user_acceptance_rate` (% of generated plans accepted without edits)
- `plan_generation_time` (target: <3s)

4.2 Feature 2: Weekly Micro-Resolution Planner

FR-2.1: Automated Weekly Planning

Priority: P0 (Core agentic behavior)
Agentic Level: High (Fully autonomous)

User Story:
As Rachel, I want the agent to automatically create my weekly plan based on how I did last week, so I always know what to focus on without having to think about it.

Acceptance Criteria:

- [ ] Every Sunday at 8 PM, automated job runs for all users with active resolutions
- [ ] Agent analyzes last week:
  - Tasks completed vs target
  - Patterns (which days user succeeded)
  - Missed tasks and reasons (from brain dumps if available)
- [ ] Agent generates next week's tasks:
  - Maintains target (e.g., 3x/week) if user succeeded
  - Adjusts difficulty if user struggled (<50% completion)
  - Avoids days/times where user consistently missed
- [ ] Sends push notification: "Your week is planned! Tap to review."
- [ ] User can view, edit, or confirm plan
- [ ] All planning logic logged to Opik

Agentic Logic:

```python
```

```python
@scheduler.scheduled_job('cron', day_of_week='sun', hour=20)
async def generate_weekly_plans():
    users = await get_users_with_active_resolutions()

    for user in users:
        with opik.track(name="weekly_planning", user_id=user.id):
            resolutions = user.active_resolutions

            for resolution in resolutions:
                # Analyze last week
                last_week = await analyze_week_performance(resolution.id)

                # Adaptive planning
                if last_week.completion_rate < 0.5:
                    # User struggled - reduce Week 2 target
                    next_week_plan = await generate_plan(
                        resolution,
                        target=resolution.target - 1,  # e.g., 2x instead of 3x
                        difficulty="easier"
                    )
                    reason = "reduced_target"
                elif last_week.completion_rate >= 0.8:
                    # User succeeded - maintain or increase
                    next_week_plan = await generate_plan(
                        resolution,
                        target=resolution.target,
                        difficulty="maintain"
                    )
                    reason = "on_track"

                # Avoid failure patterns
                failed_days = last_week.get_failed_days()  # e.g., ["Wednesday"]
                next_week_plan.avoid_days(failed_days)

                # Save and notify
                await save_weekly_plan(next_week_plan)
                await send_notification(
                    user.id,
                    f"Week {next_week_plan.week_number} is ready! {len(next_week_plan.tasks)} tasks planned.",
                    action="review_plan"
                )

                # Opik logging
                opik.log_feedback(
                    trace_id=trace_id,
                    metric_name="weekly_plan_generated",
```

```
            value=1.0,
            metadata={
                "completion_last_week": last_week.completion_rate,
                "adjustment_reason": reason,
                "tasks_planned": len(next_week_plan.tasks)
            }
        )
```

Permission Model:

- Agent generates plan autonomously
- User receives notification (not intrusive - single push)
- User can review before Monday
- If user doesn't review, plan is assumed accepted (low friction)

4.3 Feature 3: Intervention System (Simplified)

FR-3.1: Thursday Slippage Check

Priority: P0 (Outcome driver)
Agentic Level: Medium (Detects + offers options, doesn't force)

User Story:
As Rachel, when I'm falling behind on my goal, I want the agent to proactively help me get back on track before the week ends, without making me feel guilty.

Acceptance Criteria:

- [ ] Every Thursday at 7 PM, automated job checks all users
- [ ] For each active resolution:
  - Calculate: current progress vs weekly target
  - If progress <50% AND ≥2 days left in week → Trigger intervention
- [ ] Agent generates intervention message:
  - Acknowledges current state (no judgment)
  - Offers 3 clear options:
    1. Get back on track (suggests specific actions)
    2. Adjust goal (reduce target for this week)
    3. Take a break week (no guilt)
- [ ] User selects option → Agent acts accordingly
- [ ] All interventions logged to Opik with outcome

Example Intervention:

Push Notification (Thursday 7 PM):

Quick Check-In

I noticed you've done 1 workout this
week (goal: 3). Life got busy?

Tap to see options →

[In-App Card]

No judgment! Here are your options:

Option 1: Get Back on Track
- Schedule Sat 10 AM + Sun 3 PM
- Still hit your 3x goal

Option 2: Adjust This Week
- Try 2x this week instead
- Resume 3x next Monday

Option 3: Take a Break Week
- Pause until next Monday
- No streak penalty

What works for you?

Implementation:

```python
```

```python
@scheduler.scheduled_job('cron', day_of_week='thu', hour=19)
async def check_for_slippage():
    users = await get_users_with_active_resolutions()

    for user in users:
        resolutions = user.active_resolutions

        for res in resolutions:
            progress = await calculate_week_progress(res.id)
            days_left = 7 - datetime.now().weekday()  # Days until Sunday

            # Intervention criteria
            if progress.completion_rate < 0.5 and days_left >= 2:
                with opik.track(name="slippage_intervention"):
                    # Generate options
                    options = await intervention_agent.generate_options({
                        "resolution": res.title,
                        "current": progress.completed,
                        "target": progress.target,
                        "days_left": days_left,
                        "user_history": user.past_adjustments
                    })

                    # Send notification
                    await send_push(
                        user.id,
                        "Quick Check-In",
                        "Tap to see options",
                        data={"intervention_id": intervention.id}
                    )

                    # Log to Opik
                    opik.log_trace(
                        "intervention_sent",
                        metadata={
                            "resolution_id": res.id,
                            "completion_rate": progress.completion_rate,
                            "options_offered": len(options)
                        }
                    )
```

Opik Metrics:

- `intervention_effectiveness` - % of users who recover (complete ≥target after intervention)
- `option_selection_distribution` - Which options users choose most

- `false_positive_rate` - % interventions sent when user would have succeeded anyway

---

5. Permission & Autonomy Controls

5.1 Explicit Permission Model

Core Principle:
Agent acts autonomously **within user-defined boundaries**. User always has control.

Permission Levels (User Settings):

Permission Type	Default	What It Controls
Weekly Planning	ON (opt-out)	Agent generates weekly plans automatically
Interventions	ON (opt-out)	Agent sends Thursday slippage checks
Push Notifications	ON (with caps)	All proactive agent messages
Quiet Hours	10 PM - 8 AM	No notifications during these hours
Notification Frequency Cap	3/day max	Prevents spam

Settings UI:

5.2 Agent Actions Log (Transparency)

Requirement:
Every autonomous action the agent takes is logged in-app with undo option.

Example Log Entry:

```
┌──────────────────────────────────────────────────────────┐
│ Agent Actions (This Week)                                │
├──────────────────────────────────────────────────────────┤
│                                                          │
│ Sun, Jan 5 - 8:00 PM                                     │
│ 📅 Generated Week 2 Plan                                 │
│ • 3 workouts scheduled                                   │
│ • Avoided Wednesday (per your                            │
│   pattern)                                               │
│ [View Plan] [Undo]                                       │
│                                                          │
│ Thu, Jan 2 - 7:00 PM                                     │
│ ⚠️ Sent Intervention                                     │
│ • You were at 1/3 workouts                               │
│ • You chose: "Adjust to 2x"                              │
│ [View Details]                                           │
│                                                          │
└──────────────────────────────────────────────────────────┘
```

5.3 "Snooze Coach" Mode

Requirement:
User can pause all proactive agent behavior with one tap.

UI:

```
[Pause Coaching]
─────────────────────────

Pause all agent check-ins and
planning for:

○ 1 week
○ 2 weeks
● Until I resume manually

Your resolutions stay active, but
the agent won't send notifications
or create plans.

[Pause] [Cancel]
```

Implementation:

```python
```

```python
# When user pauses coaching
async def pause_coaching(user_id, duration_days=None):
    user = await get_user(user_id)
    user.coaching_paused = True
    user.pause_until = datetime.now() + timedelta(days=duration_days) if duration_days else None
    await db.save(user)

    # Disable scheduled jobs for this user
    scheduler.pause_job(f"weekly_planning_{user_id}")
    scheduler.pause_job(f"intervention_check_{user_id}")

    opik.log_feedback(
        metric_name="coaching_paused",
        value=1.0,
        metadata={"duration_days": duration_days}
    )
```

---

6. UX Design Specifications

6.1 Home Screen (Updated with Dual Intake)

```
┌─────────────────────────────────────────────────────────────┐
│  Today                                    [Profile]         │
├─────────────────────────────────────────────────────────────┤
│                                                             │
│  This Week's Focus                                          │
│                                                             │
│  Exercise 3x/week: 2/3 ✅                                   │
│  Read 12 books: 45/60 pages ⚠️                              │
│                                                             │
│  ─────────────────────────────────────────────────────      │
│                                                             │
│  Monday, Jan 6                                              │
│                                                             │
│  ☐ 30-min jog (9 AM)                                        │
│    Exercise 3x/week                                         │
│    [Start] [Reschedule]                                     │
│                                                             │
│  ☐ Read 20 pages (8 PM)                                     │
│    Read 12 books                                            │
│    [Mark Done]                                              │
│                                                             │
├─────────────────────────────────────────────────────────────┤
```

[🧠 Brain Dump] [New Resolution] ← Dual intake

7-12. [Remaining Sections - Streamlined]

Key Updates to Final Sections:

Section 8 (Data Models):

- Add `brain_dumps` table with `signals_extracted` JSONB field
- Add `agent_actions_log` table for transparency
- Add `user_preferences` table with permission settings

Section 10 (Opik Integration): Add 3 trace types:

1. `brain_dump_processing` - Signal extraction quality
2. `resolution_decomposition` - Planning accuracy
3. `weekly_planning` - Adaptation logic
4. `slippage_intervention` - Intervention effectiveness

Section 11 (Success Metrics): Outcome Metrics:

- Resolution completion rate (target: 70%)
- Weekly adherence rate (target: 75%)
- Intervention recovery rate (target: 60%)

Guardrail Metrics (NEW):

- `nudge_fatigue_rate` - % users who pause/mute notifications
- `autonomy_violation_rate` - LLM judge: % of agent messages that command vs offer choice
- `undo_action_rate` - % of agent actions user undoes (target: <5%)

Target Dashboard:

FlowBuddy Outcomes Dashboard

Outcome Metrics:
- Resolution Completion: 72% ✅ (target: 70%)
- Intervention Recovery: 65% ✅ (target: 60%)

Guardrail Metrics:
- Nudge Fatigue: 3% ✅ (target: <5%)
- Autonomy Violations: 2% ✅ (target: <5%)
- Action Undos: 4% ✅ (target: <5%)

Status: Healthy - Agent is helpful, not annoying

---

12. Implementation Plan (4 Weeks)

Week 1: Foundation + Dual Intake

- [ ] FastAPI backend with Opik SDK
- [ ] Database schema (resolutions, tasks, brain_dumps, agent_actions_log)
- [ ] Brain Dump endpoint + signal extraction (LLM)
- [ ] Resolution Setup endpoint + decomposer agent (LLM)
- [ ] React Native: Home screen, Brain Dump screen, Resolution Setup
- [ ] Opik: First 3 traces (brain_dump, resolution_decomp)

Deliverable: User can dump brain, create resolution, see Week 1 plan

---

Week 2: Weekly Planner + Permissions

- [ ] APScheduler setup for background jobs
- [ ] Weekly planning agent (Sunday 8 PM job)
- [ ] Notification system (push to mobile)
- [ ] Permission controls UI (settings screen)
- [ ] Agent Actions Log (transparency feature)
- [ ] Opik: `weekly_planning` trace + `plan_acceptance_rate` metric

Deliverable: Agent generates Week 2 plan autonomously, user can review/pause

---

Week 3: Intervention System + Guardrails

- [ ] Thursday slippage detection job

- [ ] Intervention agent (generates 3 options)
- [ ] User response handling (reschedule / adjust / break)
- [ ] Opik: `slippage_intervention` trace
- [ ] LLM-as-Judge metrics:
  - `autonomy_support_score` (does agent offer choice?)
  - `tone_warmth` (is message supportive?)
- [ ] Guardrail tracking:
  - `nudge_fatigue_rate