PICT HACK

📌 PRODUCT REQUIREMENTS DOCUMENT

Product Name: FlowState – Smart Task & Energy Manager

---

1. Product Vision

FlowState is a behavioral intelligence layer on top of a task manager that:

- Tracks work behavior in real time
- Computes a dynamic Energy Score
- Detects velocity drops and cognitive friction
- Suggests interventions
- Generates weekly analytics (Peak Focus Hours & Burnout Risk Zones)

No RAG. No fake ML.
Rule-based + behavioral analytics.

---

2. System Architecture Overview

Frontend (React)

- Task Dashboard
- Energy Level Bar
- Suggestion Panel
- Weekly Analytics Page
- Session Timer

Backend (FastAPI / Flask)

- Event Logging API
- Scoring Engine
- Intervention Engine
- Analytics Aggregator
- Baseline Calibration Module

Database (MongoDB / PostgreSQL)

- Users
- Tasks
- Sessions
- Interaction Logs
- Energy Snapshots
- Weekly Aggregates

---

3. Backend – Core Modules & Functions

MODULE 1: Event Logging Service

1.1 log_task_start()

Input

`{ user_id, task_id, task_complexity, expected_duration, timestamp }`

Process

- Create session entry if not active
- Insert task start event

Output

`{ status: "task_started" }`

1.2 log_task_end()

Input

`{ user_id, task_id, actual_duration, errors_count, idle_time, timestamp }`

Process

- Store completion metrics
- Trigger scoring engine update

Output

`{ status: "task_completed" }`

1.3 log_error_event()

Input

`{ user_id, task_id, error_type, timestamp }`

Process

- Increment error counter
- Update rolling metrics

Output

`{ status: "error_logged" }`

1.4 log_idle_event()

Input

`{ user_id, idle_duration, timestamp }`

Process

- Add to session idle total
- Update rolling window

Output

`{ status: "idle_logged" }`

MODULE 2: Feature Engineering Engine

2.1 compute_velocity()

Input

- Completed tasks in last 20 min
- Expected durations

Formula

`velocity = (actual_completion_rate / expected_rate)`

Output

`float (0-1)`

2.2 compute_idle_ratio()

`idle_ratio = idle_time / total_active_time`

Output: float (0–1)

2.3 compute_error_rate()

`error_rate = total_errors / total_interactions`

Output: float (0–1)

2.4 compute_cognitive_friction()

`friction = error_rate × task_complexity_weight`

Weights:

- Low = 1
- Medium = 1.5
- High = 2

Output: float

MODULE 3: Energy Scoring Engine

3.1 compute_energy_score()

Input

- velocity
- idle_ratio
- friction
- user_baseline

Formula

`Energy Score = (0.4 × velocity) + (0.3 × (1 − idle_ratio)) + (0.3 × (1 − friction))`

Scaled to 0–100.

Output

`{ energy_score: int, velocity, idle_ratio, friction }`

Stored in energy_snapshots table.

MODULE 4: Baseline Calibration

4.1 update_user_baseline()

Runs daily.

Input

- Last 3 days data

Process

- Compute average velocity
- Average error rate
- Average idle ratio

Store as user_baseline.

Output

`{ baseline_velocity, baseline_error_rate, baseline_idle_ratio }`

MODULE 5: Intervention Engine

5.1 evaluate_intervention()

Input

- current_energy
- velocity_trend
- error_rate
- session_duration

Logic

Rule 1:
If velocity < 85% of baseline
→ Suggest short break

Rule 2:
If error_rate > baseline + threshold
AND task_complexity == high
→ Suggest low complexity task

Rule 3:
If session_duration > 120 mins
AND energy < 50
→ Suggest extended break

Output

`{ suggestion_type: "break" | "switch_task" | "none", message: "Take a 5-minute movement break" }`

MODULE 6: Weekly Analytics Engine

Runs once daily.

6.1 compute_hourly_aggregates()

For each hour:

- avg velocity
- avg energy
- avg error_rate

Output:

`{ hour: 10, avg_velocity: 0.82, avg_energy: 74, avg_error: 0.08 }`

6.2 identify_peak_focus_hours()

Logic:

- High velocity
- Low error
- Consistent energy

Output:

`{ peak_hours: [10, 11] }`

6.3 detect_burnout_risk_zones()

Logic:

- Declining velocity trend
- Rising error trend
- Long session durations

Output:

`{ burnout_hours: [21, 22] }`

---

4. Frontend Features

4.1 Dashboard Page

Components:

- Task Queue
- Energy Bar (real-time)
- Session Timer
- Suggestion Panel

Energy Bar

Input from backend:

`{ energy_score }`

Visual states:

- 80–100 → Green
- 50–79 → Yellow
- < 50 → Red

Smooth animation.

Suggestion Panel

Receives:

`{ suggestion_type, message }`

Displays dynamic prompt.

4.2 Weekly Analytics Page

Displays:

- Peak Focus Hours (highlighted blocks)
- Burnout Risk Zones (red shaded areas)
- Hourly Energy Heatmap
- Weekly Trend Graph

---

5. Database Schema (Essential Tables)

Users

- user_id
- baseline_velocity
- baseline_error_rate
- baseline_idle_ratio

Tasks

- task_id
- user_id
- complexity
- expected_duration

Sessions

- session_id
- user_id
- start_time
- end_time

InteractionLogs

- user_id
- task_id

- event_type
- timestamp

EnergySnapshots

- user_id
- energy_score
- timestamp

---

6. Non-Functional Requirements

- Real-time updates (< 500ms latency)
- Rolling window computation
- Smooth UI transitions
- Data persistence
- Secure authentication

---

7. What You Do NOT Include

- RAG
- LLM
- Fake ML
- Synthetic training pipelines

---

8. Future Upgrade (Optional)

If data grows:

Replace rule-based scoring with:

- Gradient Boosted Regressor for energy prediction
- Classification model for burnout risk

Only when:

- > 30 days user data

- Multiple users