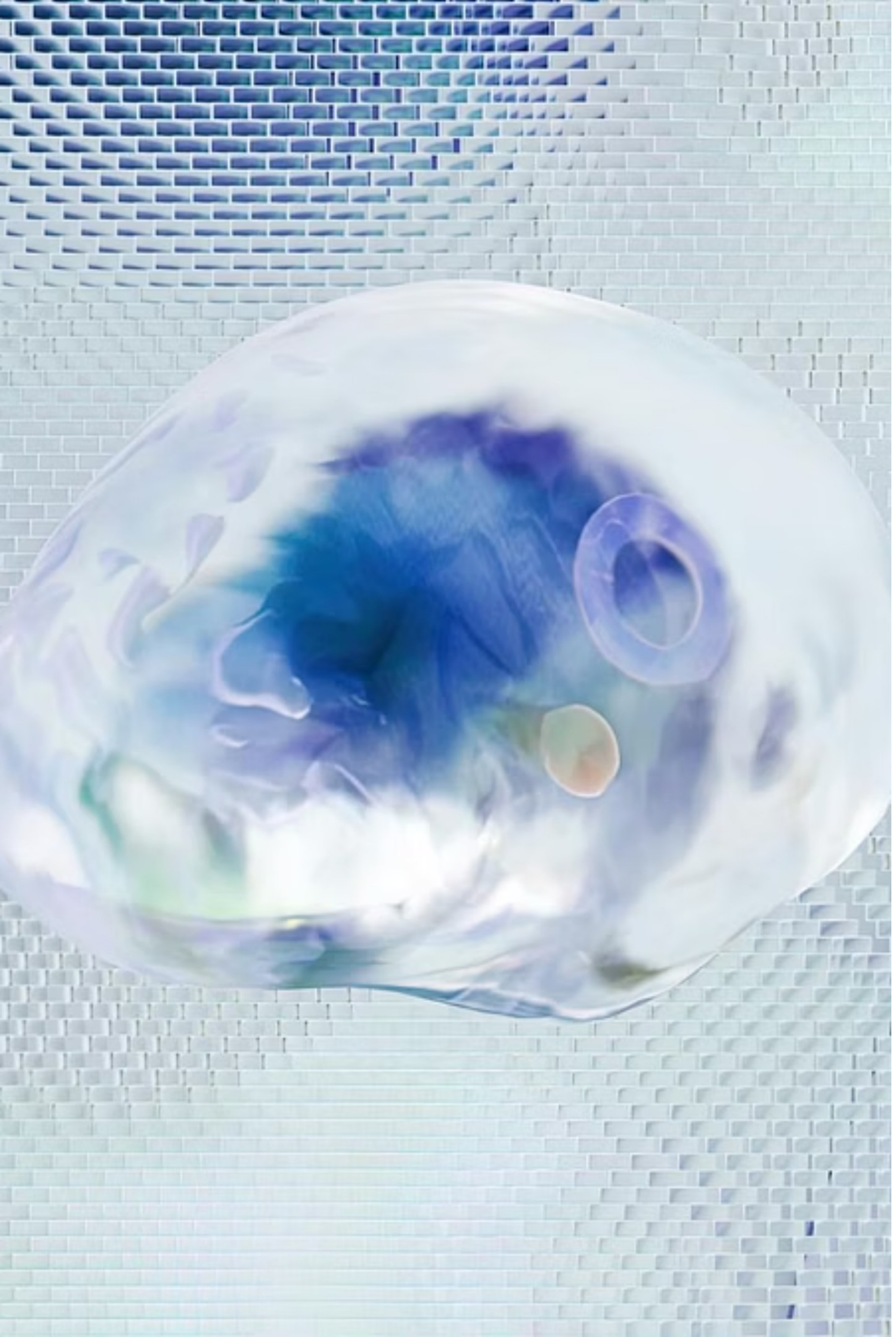


NeuroDiver-Agent:

Adapting to Neurodivergent Needs

A failure-aware execution companion that adapts to task initiation breakdown.

LSM Project Review 1

Team details:

Pratik Shah 22MID0281
Dhairya Pandey 24MID0123

Project Guide:

VINCENT HERALD WILSON

Problem Statement:

The Intention–Action Gap

Many neurodivergent individuals experience a gap between knowing what to do and actually starting it.

This isn't a lack of knowledge, but executive dysfunction.

- They know what to do.
- They want to do it.
- They do not start.
- And there are countless distractions anyway.

Why Current Tools Fail

Passive Architecture

Apps wait for user initiative, which is precisely where the struggle lies.

Rigid Alarms

Binary systems don't adapt to fatigue, overwhelm, or hyperfocus.

Context Blindness

Missed tasks are seen as discipline failures, ignoring cognitive state.

Overcomplex Assistants

Opaque intelligence reduces predictability and trust.

Current Solutions

The Market Gap

No mainstream system treats "failure-to-start" as an expected system input. This is our design opportunity.

1

Traditional Task Managers

Strong at planning, weak at execution recovery.

2

Gamified Systems

Rely on streaks and guilt, collapsing under inconsistency.

3

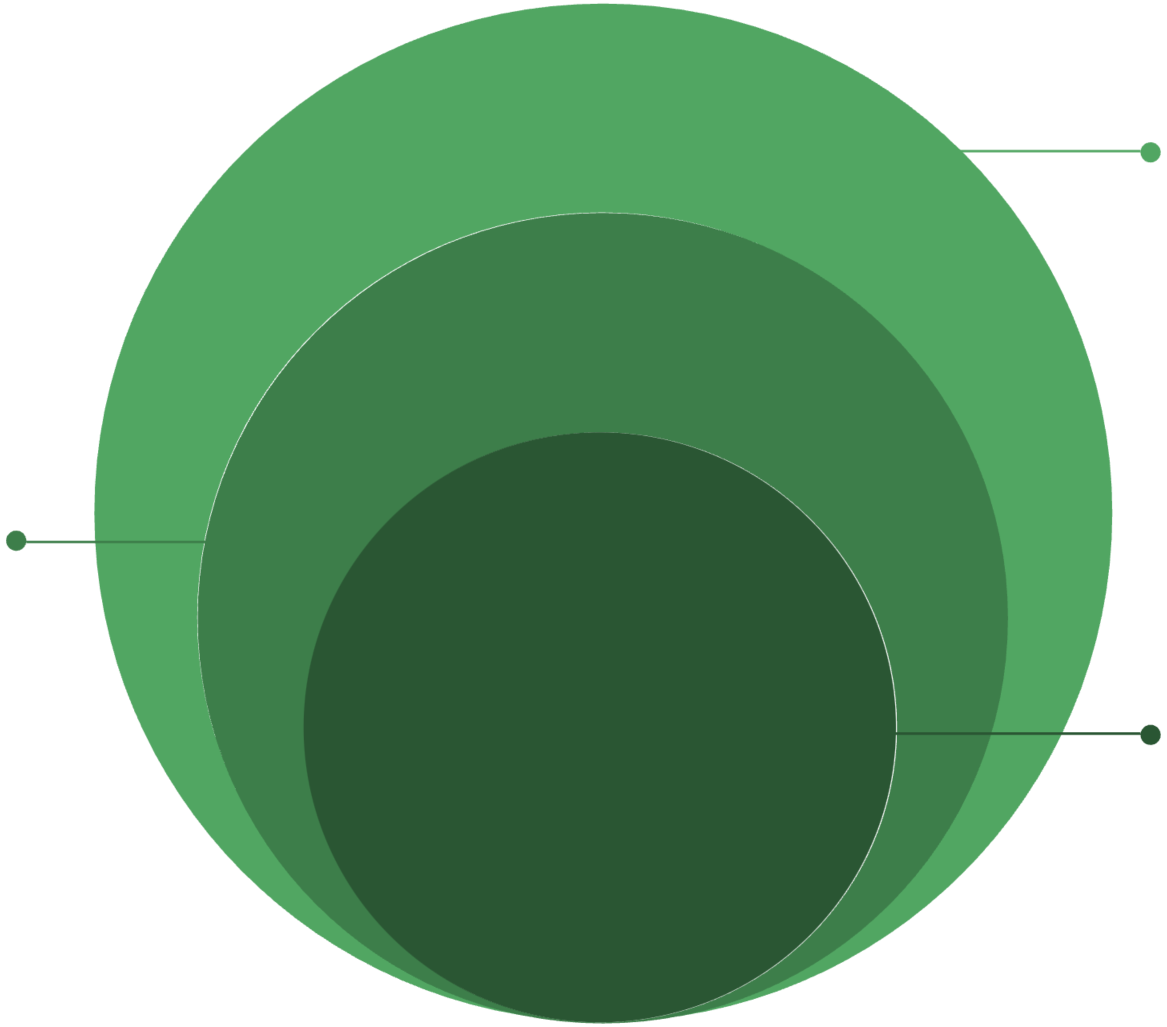
AI Planning Tools

Good at generating plans, poor at real-time behavioral adaptation.

None optimize re-entry after failure.

Our Proposal: System Architecture

Execution failure is a system event, not a personal flaw.



Reminder

Prompts that trigger tasks without blame

Task

Concrete actions where failures are handled

Goal

Desired end state guiding system behavior

This model ensures failure handling occurs only at the Task layer, allowing for adaptive recovery.

Core User Loop

01

User schedules task

02

Reminder triggers

03

User fails to start

04

System asks for state (tap-based)

05

Deterministic adaptation applied

06

User performs smallest viable action

This loop is the product, focusing on mechanical compassion, not persuasion.

OUR PROPOSAL

Deterministic Adaptation

The system adapts based on explicit execution states, with no free text or AI authority.

- **Too tired:** Downshift task + rest window
- **Overwhelmed:** Auto-split into smallest step
- **Avoiding:** Short forced timebox
- **Distracted:** Context switch prompt
- **Forgot:** Context reminder + reschedule

Controlled Expansion & Tech Stack

Future Integrations

- Energy & Attention Signals: Bias fatigue/distraction detection(social media usage).
- Friction Layer: Escalating dismissal effort.
- Focus Conditioning: Adaptive work interval expansion.
- Thought Capture: Quick context restoration.

All signals strengthen adaptation; no dashboards or scoring.

CONTROLLED EXPANSION TECH STACK

Tech Stack

- Platform: Android, Language: Kotlin
- UI: Jetpack Compose, Architecture: Unidirectional data flow
- Persistence: Room (SQLite), Scheduling: AlarmManager + WorkManager
- State Engine: Explicit Finite State Machine

SDG Alignment & Expected Outcomes

SDG 3: Good Health & Well-Being

Reduces stress, replaces shame loops, lowers cognitive load.

SDG 4: Quality Education

Improves academic re-entry, prevents deadline collapse, supports engagement.

Technical Outcomes

Functional mobile MVP, deterministic engine, extensible architecture.

Social Outcomes

Reduced shame, improved consistency, privacy-preserving support.

SDG ALIGNMENT

EXPECTED OUTCOMES

What is innovative?

Pre-failure interception

- Detect hesitation before failure using micro-signals: notification open without action, repeated screen unlocks, rapid app switching.
- Trigger state selection before the task is officially missed.

Adaptive friction gradient

- Make dismissal cost scale with repeated avoidance.
- First ignore: one tap.
- Repeated ignores: longer press, confirmation, or forced state selection.
- Friction replaces guilt as the corrective force.

State-transition memory

- Track common state sequences per task.
- Example: “Overwhelmed → Avoiding → Forgot.”
- Use this to bias future adaptations without learning models.

Hard constraint mode

- Optional mode where tasks cannot be dismissed without selecting a state.
- Useful during deadlines or exams.
- Makes the system feel authoritative without being moralizing.

Conclusion

Reframing Productivity

Most productivity systems optimize planning. NeuroDiverAgent optimizes recovery.

> Instead of asking: "Why didn't you do it?"
>
> The system asks: "What state are you in, and how should we adapt?"

This reframes productivity as a systems design problem, not a moral one.