SCRUMBOT – AI-POWERED SPRINT PLANNING ASSISTANT

FALL'25, OM VYAS (OMVYAS2) & NAKUL VASANI (NVASANI2)

Github ,Scrumbot App

PROBLEM STATEMENT AND MOTIVATION

- After sprint meetings, teams spend hours converting discussion into user stories
- Ownership decisions are subjective & uneven
- Leads to inefficiency, delays & burnout

ScrumBot solves this by automatically turning raw meeting transcripts into **structured Scrum artifacts** — complete *"As a / I want / So that"* stories with acceptance criteria and risk notes — and recommending **owners** using AI.
It blends **LLM-powered extraction** with **retrieval-augmented reasoning** over real team data (skills, capacity, history, and learning goals).

TARGET USERS & SUCCESS METRICS

- **Primary users**
- **Scrum Masters / Product Owners** who need structured stories right after meetings
- **Tech Leads & Engineering Managers** who want fair, evidence-based ownership recommendations
- **Agile Teams** seeking to reduce coordination overhead

- For Product Owners: time savings and faster documentation
- For Tech Leads: evidence-backed, fairer assignments
- For Teams: balanced workload and skill-matched growth opportunities
- For all: increased trust through transparent AI and traceable evidence

HANDLING ERRORS

- We built a lightweight **error-handling layer** to stabilize AI responses before they reach the frontend.
- **Validation:** Zod ensures schema compliance; regex catches malformed responses.
- **Reliability:** Normalized timestamps and fairness weighting reduce bias and time drift.
- **Performance:** Zustand memoization keeps UI responsive even with frequent AI calls.

Challenge	Fix / Safeguard Implemented
Non-JSON AI output	Regex filter + Zod schema validation
Missing fields	Structured type-checking before database insert
Duplicate stories	Semantic de-duplication using content embeddings
Timestamp drift	Normalized time offsets across sessions
Owner bias	Introduced fairness weighting (γ) during scoring
UI lag	Zustand memoization for reactive state management

DESIGN DECISIONS & AI MODELS

Tech Stack Overview

- **Frontend:** Next.js 15 + TypeScript + Tailwind + shadcn/ui → modern, responsive UI
- **State Management:** Zustand → lightweight global state for step-by-step sprint flow
- **Backend:** Next.js API Routes + Supabase (Postgres + Row-Level Security)
- **Storage:** Supabase tables for team skills, capacity, preferences, and story history
- **Integration:** Groq API for ultra-low-latency model inference

Purpose	Model	Why This Model
Story Extraction	**Llama 3.1 8B Instant (Groq)**	Fast, deterministic JSON generation for “As a / I want / So that” stories
Owner Ranking	**Llama 4 Maverick 17B Instruct (Groq)**	Handles numeric reasoning and multi-factor scoring (competence + availability + growth + continuity)

CHALLENGES & SOLUTIONS

Challenge	Solution / Fix Implemented
Parsing LLM JSON	Regex sanitation + Zod schema validation
Latency on large files	Chunked data processing + streamed UI rendering
Fair assignment logic	Growth & continuity weights (γ, δ) for balanced distribution
Supabase RLS issues	Moved server routes to **service role** with controlled permissions

- We encountered challenges across data parsing, scalability, and access control.
- Implemented **schema validation** to stabilize LLM output, and **chunked streaming** to minimize latency.
- Designed **fairness weights** to ensure balanced task assignment.
- Solved **Supabase permission conflicts** by shifting routes to the service role layer.

SAFETY, PRIVACY, AND RELIABILITY

- Our system enforces **row-level isolation** via Supabase RLS for per-user data access.
- **Secrets** are stored securely on the backend; no API keys are ever exposed client-side.
- We delete all **raw text data** after it's processed to prevent accidental data leakage.
- A **synthetic demo mode** ensures safe public demos.
- The **"Lock Sprint"** feature enables transparent, tamper-proof audit trails.

Aspect	Implementation / Safeguard
Data Access Control	Supabase Row-Level Security (RLS) enforced on all tables
Secret Management	API keys (e.g., `GROQ_API_KEY`, `SERVICE_ROLE_KEY`) stored securely server-side
Data Retention	No raw transcripts retained after post-processing
Demo Mode	Uses fully synthetic, non-PII data for safe testing
Audit Integrity	"Lock Sprint" creates immutable exports for compliance tracking