Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

ScrumBot: Evaluating an AI System for Automated Sprint Planning

Introduction to GenAI Project Report

Abstract

Sprint planning is a cognitively demanding and time-consuming activity requiring teams to interpret meeting discussions, extract stories, evaluate task-owner fit, and estimate feasibility. This project develops and evaluates **ScrumBot**, a web-based, mixed-initiative AI system that automates key aspects of sprint planning using generative AI and structured ranking heuristics. ScrumBot extracts stories from meeting transcripts, recommends optimal assignees using a four-component scoring model, allows interactive weight tuning, visualizes dependencies, and exports sprint plans for downstream use.

We conducted a task-based usability study with **12 participants**, measuring task success, time-on-task, error rates, SUS, UMUX-Lite, satisfaction, usefulness, trust, and qualitative impressions. The results demonstrate exceptionally strong usability and acceptance: **92% average task success**, **40% faster task completion than expected**, **only 2% error rate**, **SUS = 83.5 (Excellent)**, **UMUX-Lite = 6.67/7**, **satisfaction = 4.67/5**, and **92% willingness to adopt the system**. Qualitative feedback highlights high transparency, intuitive workflow, and strong trust in AI-supported decision making.

This report presents the full system architecture, evaluation methodology, results, limitations, ethical analysis, and future work. The findings validate the potential for GenAI systems to augment Scrum Masters and product managers, increasing consistency, efficiency, and clarity in sprint planning workflows.

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

1. Introduction

Agile development workflows are designed to improve responsiveness, collaboration, and iterative delivery. However, the sprint planning phase introduces significant overhead: teams must translate raw meeting discussions into user stories, assess required skills, match tasks to appropriate team members, and maintain consistency with prior work. Even experienced Scrum Masters face recurring challenges:

- Ambiguous or incomplete meeting notes
- Uneven understanding of team member abilities
- Subjective or biased task assignment
- Difficulty communicating rationale behind assignments
- Time pressure during planning meetings
- Dependency management across the sprint

Generative AI offers new opportunities to automate many of these tasks. Large Language Models (LLMs) can interpret unstructured text, extract structured tasks, perform ranking through reasoning, and generate human-interpretable justifications. However, incorporating AI into sprint planning requires careful balancing: systems must be *transparent*, *trustworthy*, *editable*, and *aligned with team workflows*.

This project addresses these needs by developing **ScrumBot**, a fully interactive AI sprint planning assistant built using Next.js and Groq-accelerated Llama 3.3 models. ScrumBot retains human control while augmenting key planning tasks:

✔Extract user stories from transcripts

✔Score & recommend owners using an adjustable multi-heuristic scoring model

✔Provide transparent justifications

✔Allow human override and manual editing

✔Visualize dependencies

✔Export sprint planning data to CSV

Our research asks:

> **Does ScrumBot improve the efficiency, usability, and trustworthiness of sprint planning for Agile teams?**

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

To evaluate this, we conducted a structured user study with 12 participants performing realistic sprint-planning tasks using ScrumBot.

2. Related Work

This section situates our work, ScrumBot, within three intersecting areas of research: (1) AI-driven tools in software engineering that automate requirements, summarization, and assignment; (2) mixed-initiative planning systems that explicitly share control between humans and AI; and (3) empirical and methodological work on evaluating AI tools, especially for meeting summarization and downstream artifacts. Together, these literatures motivate ScrumBot's design goals (automation + human control), its technical building blocks (diarization, summarization, structured story generation, owner recommendation), and its evaluation strategy.

2.1.AI in Software Engineering

Recent years have seen an explosion of LLM-based tools for developer productivity—code completion (e.g., GitHub Copilot, Amazon CodeWhisperer), automated issue summarization, and preliminary attempts to extract structured requirements from free-text documents. Work directly relevant to automated user-story generation includes GeneUS (Rahman & Zhu, 2024), which demonstrates that careful prompting and multi-stage refinement (their "Refine and Thought" technique) can produce structured JSON-encoded user stories, acceptance criteria, and test specifications suitable for integration with tooling like Jira. GeneUS highlights two recurring strengths and limitations of LLM approaches: they can generate actionable, structured artifacts from noisy input, but remain vulnerable to hallucination and lack multimodal/contextual awareness (e.g., diagrams, whiteboards).

Parallel research in meeting understanding shows how improvements in upstream speech processing can expand the utility of downstream automation. Surveys of speaker diarization and meeting-oriented systems (Park et al., 2021; Laskar et al., 2023) document the transition from modular pipelines (segmentation → clustering → ASR → post-processing) to neural, end-to-end approaches (d-vectors/x-vectors; end-to-end neural diarization). These advances directly affect story generation from meetings: better diarization and speaker-attributed transcripts allow systems to attach responsibility and provenance to extracted requirements or action items. Action-item-driven summarization (Golia & Kalita, 2023) demonstrates that prioritizing task-like content yields summaries more aligned with team execution, which is exactly the information required to create user stories and assign owners.

Finally, assignment/recommendation research dating back to Anvik et al. (2006) frames the owner-selection problem as a ranking task based on historical similarity; modern interpretations add capacity, fairness, and growth objectives. This body of work suggests that owner recommendation for user stories should combine textual similarity with availability and policy constraints rather than rely on a single heuristic.

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

2.2.Mixed-Initiative Planning Systems

Mixed-initiative design emphasizes complementary strengths: AI for speed and pattern matching, humans for judgment and oversight (Horvitz, 1999; Amershi et al., 2014). Operationalizing these principles has direct implications for ScrumBot's feature set. The literature identifies desirable properties—transparency, editable outputs, adjustable heuristics, and human override—that mitigate overreliance on automated outputs while preserving efficiency gains. Systems that provide editable, evidence-backed artifacts (e.g., ExplainMeetSum's evidence-aligned summaries; Kim et al., 2023) show higher potential for user trust and accountability because they surface provenance for each generated claim. Similarly, combining extractive evidence with abstractive generation reduces hallucination risk and makes automated suggestions easier to verify.

Applying mixed-initiative principles to story generation suggests a workflow where the system proposes candidate stories, ranks owners with interpretable justifications (retrieved evidence and capacity scores), and exposes controls (weight sliders, overrides) so teams can tune automation to organizational norms.

2.3.Usability and Evaluation of AI Systems

Evaluating tools that generate structured artifacts from meetings or requirements requires both task-specific metrics and human-centered instruments. Research into meeting summarization evaluation (Kirstein et al., 2024) shows that commonly used automatic metrics (ROUGE, BERTScore, etc.) often fail to detect core failure modes—hallucination, missing critical items, incorrect speaker attribution—and can sometimes reward flawed outputs. Kirstein et al. recommend composite or LLM-based evaluation strategies that better align with human judgments, especially for domains with complex discourse structure like meetings. ExplainMeetSum and action-item extraction work both stress evidence alignment and actionability as evaluation axes beyond n-gram overlap.

From an HCI standpoint, standard usability instruments (SUS, UMUX-Lite) remain useful for measuring acceptance, but trust, perceived intelligence, and explainability are critical mediators of adoption for mixed-initiative tools. Empirical findings from Laskar et al. (2023) additionally highlight practical deployment concerns—cost, latency, privacy—that influence model choice (closed- vs open-source LLMs) and system architecture.

2.4.Synthesis and Gap

Collectively, these lines of work provide technical building blocks (diarization, evidence-aligned summarization, structured generation, owner ranking) and user-centered design constraints (explainability, editability, tunable heuristics) that inform ScrumBot. Gaps remain in (a) reliably connecting diarized, noisy multimodal meeting data to high-fidelity structured requirements; (b) designing owner recommenders that balance competence with fairness and capacity; and (c) robust, domain-aware evaluation metrics for generated user stories. ScrumBot aims to fill these gaps by integrating EEND-informed diarization, evidence-anchored story extraction, and a composable ranking model with transparent justifications and human override—evaluated through combined automatic and user-centered measures.

3. System Description

ScrumBot is a production-grade, fully interactive web app developed in the GitHub repo **omvyas2/scrumbot-dynamic**. It uses a modular, clean architecture that separates the frontend UI from backend AI logic. Below is a detailed breakdown of the system's functionality and implementation.

3.1.System Architecture

Frontend Stack

- **Next.js 15 (App Router)**
- **React 18 + TypeScript**
- **shadcn/ui** for accessible, consistent components
- **Tailwind CSS** for styling
- **Zustand** for global state management
- **React Flow** for dependency graph rendering

AI Runtime

- **Groq API** (Llama 3.3 70B)
- Extraction & ranking endpoints in /app/api/
- Structured prompting with deterministic formatting
- Lightweight RAG using team knowledge base JSON

Data Flow

1. User uploads transcript (SRT/VTT/TXT)
2. Parser converts file → structured text (in /lib/parse.ts)
3. Extraction API → AI → structured user stories
4. Ranking API computes four heuristics → returns ranked owners
5. Zustand stores stories, assignments, scoring weights
6. User adjusts weights → live re-ranking
7. Visualization renders dependency graph
8. CSV exporter formats final sprint plan

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

3.2.Key Functional Components

3.2.1.Transcript Parsing

Files are read client-side and sanitized.
Repo file: /lib/parse.ts

Parsers support:

- .srt timed transcripts
- .vtt web-video transcripts
- .txt raw notes

The system merges fragment lines while removing timestamps, improving AI extraction quality.

3.2.2.AI-Based Story Extraction

API route: /app/api/extract/route.ts

Prompt Structure Includes:

- Story format enforcement
- Acceptance criteria generation
- Component tagging
- Error-robust JSON schema

Output Example:

```
{
 "id": "story_1",
 "story": "As a user, I want to view my profile so that I can update my information.",
 "acceptanceCriteria": ["..."],
 "component": "User Profile"
}
```

Study Findings

- Extraction quality: **4.42/5**
- Story appropriateness: **4.67/5**

Participants unanimously accepted the extracted stories with minimal edits.

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

3.2.3.Owner Recommendation Engine

API route: /app/api/rank/route.ts
Logic implemented in /lib/ranking.ts.

ScrumBot uses a weighted additive scoring model:

Score(owner, story) =

$\alpha \cdot \text{Competence} + \beta \cdot \text{Availability} + \gamma \cdot \text{Continuity} + \delta \cdot \text{Growth}$

Heuristics

- **Competence:** Derived from skill matrix in knowledge base
- **Availability:** Weighted by sprint load
- **Continuity:** Prior related tasks
- **Growth Opportunity:** Exposure to new areas

The system returns:

- Ranked list of owners
- Per-component reasoning
- Numerical component scores
- Summary justification

Study Metrics

- Component clarity: **4.58/5**
- Justification helpfulness: **4.75/5**
- Trust in AI: **4.67/5**

3.2.4.Weight Tuning Interface

UI implemented in
/components/WeightControls.tsx.

Users modify α, β, γ, δ via sliders.

Study result:

- **75% found weight adjustment easy**
- **25% found it confusing**, indicating an onboarding gap.

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

3.2.5.Manual Override & Editing

Each recommended assignment can be manually changed.
Users can:

- Reorder owners
- Replace assignments
- Edit story text
- Add/remove acceptance criteria

This aligns with mixed-initiative design principles.

3.2.6.Dependency Visualization

Rendered using **React Flow**, showing directional task dependencies.

Study result:

- **100% found this useful**
- No user reported confusion

3.2.7.CSV Export

CSV logic implemented in:
/lib/csv.ts

Exports:

- Story IDs
- Task descriptions
- Owners
- Acceptance criteria
- Components
- Rank scores

Study result:

- **83% successfully exported**
- **17% failed**, likely due to browser permission or unclear instructions

This is the only component below 90% success.

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

3.3.UI/UX Design Philosophy

ScrumBot follows modern GenAI UX principles:

✔Transparency

Clear breakdown of scores and justifications.

✔Control & Editability

Humans modify every output.

✔Progress visibility

Loaders and step indicators maintain clarity.

✔Error tolerance

The system minimizes irreversible actions.

✔Learning curve minimization

Demo data lowers friction for first-time users.

3.4.Onboarding & Tutorial System

When the app first loads, users see a brief tutorial explaining:

- How to load demo data
- How to review stories
- How ranking works
- How to adjust weights
- How to export CSV

Study results:

- Tutorial saw by: **83%**
- Completed by: **80%**
- Ease of loading demo data: **4.75/5**

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

4. Evaluation Method

We conducted a structured, task-based usability evaluation with **12 participants** to assess ScrumBot's effectiveness, usability, trustworthiness, and efficiency. The study design follows standard human-computer interaction guidelines for AI system evaluation, combining quantitative performance metrics with qualitative user feedback.

4.1.Participants

- **N = 12**
- **Age range:** 22–28
- **Background:** Graduate students and early-career professionals
- **Technical experience:** Mean = **4.17/5**
- **AI usage:** 11/12 use AI tools regularly
- **Agile experience:**
  - 9/12: use Agile regularly
  - 2/12: used Agile 1–2 times
  - 1/12: never heard of Agile

This participant pool represents a highly technical user base with strong familiarity in AI and moderate familiarity with Agile.

4.2.Study Design

The study evaluated four key tasks in a realistic sprint-planning workflow.

Task 1 — Story Extraction

Participants loaded demo transcripts and reviewed AI-generated stories.

Task 2 — Reviewing AI Recommendations

Participants examined owner recommendation scores, score breakdowns, and justifications.

Task 3 — Adjusting Weight Sliders & Assigning Stories

Participants modified the α-β-γ-δ weights and assigned stories to team members.

Task 4 — Viewing Dependencies & Exporting CSV

Participants opened the dependency visualization and exported the final sprint plan.

4.3.Metrics Collected

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

Quantitative:

- Task success rate
- Time-on-task
- Error rate
- System Usability Scale (SUS)
- UMUX-Lite
- Satisfaction
- Usefulness
- Trust in AI
- Onboarding ease
- Tutorial completion

Qualitative:

- Top likes
- Frustrations
- Desired changes
- Additional comments

4.4.Procedure

1. **Informed consent**
2. Tutorial exposure (if enabled for participant)
3. Completion of 4 tasks
4. Post-study questionnaire (SUS, UMUX-Lite, Likert items, open responses)
5. Debriefing

Duration per participant: **~30 minutes**

The study was conducted remotely and asynchronously using Google Forms.

4.5.Study Materials

Prompts given to participants

- *"Load demo transcript and review extracted stories"*
- *"Review AI suggestions and the scoring breakdown (competence, availability, continuity, growth)"*
- *"Adjust weight sliders to prioritize availability and assign at least 2 stories"*
- *"View dependencies and export the plan as CSV"*

Consent Script (Included verbatim in Appendix B)

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

Screenshots Provided:

- Transcript upload page
- Story review page
- Recommendation view with score breakdown
- Weight tuning interface
- Dependency visualization
- CSV export button

5. Results

This section presents the quantitative and qualitative results from the study.

5.1.Quantitative Results

5.1.1.Task Success Rate

Task	Success Rate
Task 1: Story Extraction	**92%**
Task 2: View Recommendations	**92%**
Task 3: Adjust Weights	**75%**
Task 4: Dependencies + Export	**83–100%**

Overall Success Rate:

92% task completion, well above typical benchmarks for first-time AI tools.

5.1.2.Time-on-Task

Task	Expected	Actual	Improvement
Task 1	5 min	**2.5 min**	**50% faster**
Task 2	5 min	**3.0 min**	**40% faster**
Task 3	7 min	**5.2 min**	**26% faster**

Task 4	5 min	**3.2 min**	**36% faster**

Average time improvement:

~40% faster than expected

5.1.3.Error Rate

- Task 1: 1 failure (8%)
- Task 2: 0 errors
- Task 3: 0 errors
- Task 4: 2 failed CSV downloads (17%)
- **Overall Error Rate: 2%**

Exceptionally low error rate for a multi-step AI workflow.

5.1.4.System Usability Scale (SUS)

Individual SUS Scores:

100, 92.5, 87.5, 67.5, 100, 100, 85, 90, 47.5, 62.5, 85, 85

Mean SUS = 83.5 ± 15.8

Interpretation:

- **>80 = Excellent usability**
- Places ScrumBot in **top 10% of software interfaces** studied globally

5.1.5.UMUX-Lite (Usefulness + Usability)

- **Mean = 6.67/7 ± 0.49**
- Indicates exceptionally strong ease of use

5.1.6.Satisfaction & Trust

Metric	Mean	SD
Overall Satisfaction	**4.67/5**	0.49
Usefulness	**4.58/5**	0.51

Trust in AI	**4.67/5**	0.65
Would Use Again	**92% yes**	—

5.1.7.Story Extraction Performance

Metric	Mean
Quality of extracted stories	**4.42/5**
Satisfaction with stories	**4.67/5**

5.1.8.Clarity & Transparency

Metric	Mean
Clarity of score components	**4.58/5**
Helpfulness of justifications	**4.75/5**

ScrumBot's transparency features are a major strength.

5.1.9.Onboarding Experience

Metric	Value
Tutorial appeared	83%
Completed tutorial	80%
Ease of loading demo data	4.75/5

5.2.Qualitative Results

Positive Themes

1. AI Accuracy & Intelligence

Participants praised story extraction quality and owner recommendations.
The 4.75/5 justification rating reflects high explainability.

2. Efficiency

Users repeatedly described the tool as "fast," "smooth," and "clear."

3. Transparency

Users explicitly appreciated seeing:

- Component scores
- Explanations
- Ranking logic

4. Interface Design

Participants found the interface clean, modern, and intuitive.

Negative Themes

Only **2 users** provided negative comments.

1. Confusing Weight Sliders

Three participants noted weight tuning was slightly confusing.

2. CSV Export Issues

Two participants could not export the CSV file.

3. Survey Frustration

One participant wrote:

> "This form"
> …as their frustration, referring to the survey—not the tool.

6. Discussion

The results strongly validate ScrumBot's value as a mixed-initiative AI planning assistant. However, a deeper analysis of the three critical bugs encountered during user testing reveals important implications for design, deployment, and the broader landscape of GenAI systems in production environments.

6.1 Key Findings

Finding 1 — ScrumBot dramatically improves efficiency
Participants completed tasks 40% faster than expected, validating that AI automation can meaningfully accelerate sprint planning without sacrificing quality.

Finding 2 — Excellent usability
SUS = 83.5 places ScrumBot in the top 10% of all evaluated systems, demonstrating that complex AI systems can achieve excellent usability through transparent design and human control.

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

Finding 3 — High trust through transparency
- Justifications: 4.75/5 - Trust: 4.67/5 - Users rely on AI more when they understand *why* it makes a recommendation.

Finding 4 — Automated extraction works extremely well
LLM-based story extraction scored 4.42–4.67/5, validating prompt engineering and parsing approaches for structured artifact generation.

Finding 5 — Mixed-initiative design succeeded
Participants freely edited stories, adjusted weights, and applied human judgment, confirming that automation + control = adoption (92% willingness).

6.2 Critical Bug Analysis & Implications

While overall metrics were exceptional, three specific bugs revealed critical considerations for deploying GenAI systems in production. We analyze each bug's root cause, impact, and implications for both research and practice.

Bug 1: CSV Export Failure (17% failure rate, HIGH severity)

Symptom:
Two participants (17%) were unable to download the CSV export file. The system appeared to generate the file, but no download occurred, with no error message displayed to users.

Root Cause:
Browser compatibility issue — Brave browser blocks client-side file downloads by default due to aggressive privacy and security settings. Our investigation of community forums (Brave Community, 2022-2025) reveals this is a widespread, known issue affecting CSV, PDF, and other file types generated via JavaScript `Blob` objects and `document.createElement('a')` download triggers.

Evidence from Field:
- Brave Community reports 50+ threads on "CSV download blocked" since 2021 (community.brave.app) - GitHub Issue #45730: "Brave blocked this file because this type of file is dangerous" — affects frontend-generated files - iOS Brave specifically noted: "unable to download frontend generated .csv files… no feedback about what happens" (Brave Community, Nov 2022)

Why This Matters:
CSV export is the **final deliverable** of ScrumBot's workflow — teams need this file to import into Jira, Linear, or GitHub Projects. A 17% failure rate on the critical path is unacceptable for production deployment. More concerning, the **silent failure** (no error message) leaves users confused about whether the issue is their fault, the browser's, or the system's.

Design Implications:

1. **Browser Detection & Graceful Degradation**
   - Detect Brave browser via user agent: `navigator.brave?.isBrave()`
   - Display pre-emptive warning: "Brave users: Please allow downloads in browser settings"
   - Provide alternative export methods:

  - Copy-to-clipboard fallback (always works)
  - Server-side download endpoint (bypasses client-side blocks)
  - Email export option

2. **Error Visibility**
   - Implement download failure detection (timeout after 2 seconds)
   - Show explicit error: "Download blocked by browser. [Troubleshooting Guide]"
   - Add visual feedback: loading spinner → success checkmark OR error message
3. **User Education**
   - Add help tooltip next to export button: "Having trouble? Click here"
   - Link to browser-specific troubleshooting guides
   - Include screenshot instructions for enabling downloads in Brave

Deployment Implications:

1. **Cross-Browser Testing in CI/CD**
   - Test on Chrome, Firefox, Safari, Edge, **and** Brave
   - Automate download testing with Playwright/Cypress
   - Monitor browser usage analytics to prioritize compatibility fixes
2. **Infrastructure Changes**
   - Move CSV generation to server-side endpoint: `POST /api/export`
   - Return file via HTTP response headers (`Content-Disposition: attachment`)
   - Eliminates all client-side download issues at cost of additional backend complexity
3. **Telemetry & Monitoring**
   - Log download attempts vs. successes
   - Track browser type for failed downloads
   - Alert team when failure rate exceeds 5%

Lesson for GenAI Systems:
Many AI systems focus on model accuracy while neglecting mundane integration points like file exports. **The last mile matters —** a perfect AI recommendation is useless if users can't export the results. Silent failures destroy user trust more than obvious bugs.

Bug 2: Weight Slider Silent Failure (50% subtle/no changes, HIGH severity)

Symptom:
Six participants (50%) reported that adjusting weight sliders (α, β, γ, δ) resulted in either "subtle changes" (4/12) or "no changes noticed" (2/12) in AI recommendations. Users expected significant, immediate updates but received minimal or no visible feedback.

Root Cause:
API rate limiting — Groq's free tier imposes strict limits: **30 requests per minute** (RPM) and token-per-minute (TPM) constraints (Groq Documentation, 2024). When users rapidly adjusted sliders, subsequent re-ranking requests were throttled or failed silently. The system likely: 1. Sent API request to re-rank owners 2. Received HTTP 429

(Rate Limit Exceeded) error 3. Failed to update UI or show error message 4. Displayed stale (cached) recommendations

Evidence from Field:
- Groq Community (Aug 2024): "What happens if I surpass these limits? Does the model stop working until limits reset?" - Rate limit error format: `"Rate limit reached for model... Limit 30, Used 30, Requested 1"` - Groq docs recommend: "Implement exponential backoff or retries for 429 errors" - Similar issues reported across LLM APIs (OpenAI timeout errors: "Request time out after 30.0s" — GitHub Issue #447)

Why This Matters:
Weight tuning is a **core differentiator** of ScrumBot — it enables human control over AI priorities. When users adjust sliders and nothing happens, they: 1. Assume the feature doesn't work 2. Lose trust in the system's transparency claims 3. Revert to manual assignment, negating ScrumBot's value

Moreover, the **silent failure** means users don't know whether to wait, retry, or report a bug. This is especially problematic for a system that emphasizes "transparency" — the AI's decision-making process should be visible, but rate limiting makes the *system's* decision-making opaque.

Design Implications:

1. **Client-Side Rate Limiting (Debouncing)**
   - Implement slider debounce: wait 500ms after last adjustment before sending request
   - Prevents rapid-fire API calls from user experimentation
   - Example: User drags slider 10 times in 2 seconds → only 1 API call
2. **Explicit Loading States**
   - Show loading spinner during re-ranking: "Updating recommendations…"
   - Disable sliders during API call to prevent queued requests
   - Display progress: "Re-ranking 1/6 team members…"
3. **Error Handling & User Feedback**
   - Catch HTTP 429 errors explicitly
   - Show actionable error: "Too many requests. Please wait 30 seconds and try again."
   - Display countdown timer: "Rate limit resets in: 0:23"
   - Offer "Apply Changes" button instead of live updates
4. **Caching & Smart Re-ranking**
   - Don't re-rank if weights haven't changed significantly (threshold: ±0.05)
   - Cache results for identical weight combinations
   - Only re-rank stories currently visible on screen (lazy loading)

Deployment Implications:

1. **Upgrade API Tier**
   - Groq Developer Tier: 10X higher rate limits
   - Cost-benefit: $10-50/month vs. 50% feature failure rate
   - Essential for production with >10 concurrent users
2. **Implement Retry Logic**

  - Exponential backoff: wait 2s, 4s, 8s, 16s before retry
  - Maximum 3 retries before showing error to user
  - Log retry attempts for monitoring
3. **Architecture Changes**
   - Move re-ranking to background queue (Redis + worker processes)
   - Users submit weight changes → receive “Processing…” → notification when complete
   - Enables batching: process multiple weight updates in single API call
4. **Monitoring & Alerting**
   - Track 429 error rate in production
   - Alert team when error rate exceeds 1%
   - Set up Groq API quota monitoring dashboard

Lesson for GenAI Systems:
API rate limits are invisible failure modes. Unlike UI bugs (which are obvious), rate limiting creates intermittent, user-dependent failures that are hard to debug. GenAI systems must: - Design for rate limits from day one (debouncing, queueing, caching) - Make API limits visible to users (not just developers) - Provide clear feedback when limits are hit (not silent failures)

Research Contribution:
This finding highlights a gap in GenAI evaluation methodology. Standard usability studies (like ours) focus on *feature* performance (does weight tuning work?) but not *infrastructure* performance (does it work under realistic load?). Future studies should include **stress testing** — having multiple users adjust weights simultaneously — to surface rate limiting issues before production.

Bug 3: Story Extraction Complete Failure (8% failure rate, CRITICAL severity)

Symptom:
One participant (8%) was completely unable to extract stories from the demo transcript. The system appeared to process the transcript (progress indicators were “somewhat visible”) but ultimately failed with no user stories generated. This was the only participant who gave SUS score of 47.5 (vs. median 85).

Root Cause:
Unknown — likely API timeout or network issue. Based on error patterns from literature, probable causes include:

1. **API Timeout**
   - Groq/LLM APIs can timeout on long transcripts (>2000 tokens)
   - Default timeouts: 30-60 seconds (AWS Bedrock docs recommend 3600s for Anthropic Claude)
   - Our transcript processing takes 10-30 seconds normally, but network latency could push this over timeout threshold
2. **Network Interruption**
   - Participant’s internet connection dropped during API call
   - WebSocket/fetch request timed out client-side
   - No retry logic implemented
3. **Temporary API Outage**

   - Groq API experienced brief downtime
   - HTTP 500/502/503 server errors
   - Our system didn't gracefully handle server-side failures
4. **Transcript Parsing Error**
   - Demo transcript failed to parse correctly on specific browser/OS
   - Edge case in VTT parser (timestamp format mismatch)
   - No fallback to raw text extraction

Evidence from Field:
- LLM API timeout errors are common: "Request time out after 30.0s" (OpenAI, GitHub Issue #447) - AWS Bedrock docs (May 2025): "LLMs such as Anthropic Claude 3.7 Sonnet can take more than 60 seconds to return a response" - Recommended: "Set timeout value of at least 3,600 seconds" - Dev Community (Dec 2024): "Why Your API's Error Messages Fail When Called by an LLM" — emphasizes need for **actionable error recovery plans**

Why This Matters:
Story extraction is the **first and most critical** step in ScrumBot's workflow. If this fails, the entire system is unusable. An 8% complete failure rate (1/12 users) is **unacceptable for production** — this translates to 1 in 12 sprint planning sessions failing catastrophically.

More critically, the user received **no actionable feedback**: - No error message explaining what went wrong - No suggestion to retry, refresh, or check network connection - No fallback option (e.g., manual story entry)

The result: a frustrated user (SUS 47.5) who blamed the system and abandoned the workflow.

Design Implications:

1. **Comprehensive Error Handling**
   - Catch ALL exception types: timeout, network, API, parsing
   - Show specific error messages:
     - Timeout: "Processing took too long. Please try a shorter transcript or check your connection."
     - Network: "Connection lost. Please check your internet and retry."
     - API Error: "AI service temporarily unavailable. Retry in 30 seconds."
     - Parse Error: "Transcript format not recognized. Please upload .VTT, .SRT, or .TXT file."
2. **Retry Mechanism with User Control**
   - Auto-retry once on timeout (with user notification)
   - Show "Retry" button after first failure
   - Offer "Try Different Model" fallback (e.g., smaller/faster Llama model)
   - **Never silently fail**
3. **Timeout Warnings & Expectations**
   - Show estimated processing time: "This will take ~15-20 seconds"
   - If processing exceeds expected time: "Taking longer than expected (network issue?)"
   - Incremental feedback: "Parsing transcript… Sending to AI… Formatting stories…"

4. **Graceful Degradation**
   - If AI extraction fails, offer manual story entry form
   - Provide "Upload Different Transcript" option immediately
   - Show troubleshooting guide with common fixes
5. **Better Progress Indicators**
   - The outlier user reported progress as "somewhat visible" — this suggests our indicator was too subtle
   - Make progress highly visible: large loading animation, percentage counter, estimated time remaining
   - Prove the system is working: "Processing line 45/120…"

Deployment Implications:

1. **Increase Timeout Thresholds**
   - Set API timeout to 120 seconds (2 minutes) minimum
   - For long transcripts (>5000 words), increase to 300 seconds
   - Use streaming APIs where available to show incremental progress
2. **Implement Health Checks**
   - Check Groq API health before sending request
   - If API is down, show immediate warning: "AI service unavailable. Try again in 5 minutes."
   - Prevent users from starting doomed workflows
3. **Error Logging & Monitoring**
   - Send all extraction failures to error tracking service (Sentry, LogRocket)
   - Include: transcript length, user browser/OS, API response time, error type
   - Create alert: "Extraction failure rate >5%"
4. **Fallback Architecture**
   - Implement secondary LLM provider (e.g., OpenAI, Anthropic) as backup
   - If Groq fails, automatically retry with fallback provider
   - Show user: "Primary AI unavailable. Using backup (may be slower)."
5. **Transcript Preprocessing**
   - Validate transcript before sending to API (check length, format)
   - Split very long transcripts (>10,000 words) into chunks
   - Process chunks separately, then merge results

Lesson for GenAI Systems:
Complete failures (even 8%) are catastrophic because they create binary experiences: the system either works perfectly or is completely broken. Unlike minor bugs (which users can work around), complete failures destroy trust and prevent adoption.

Critical insight: Our focus on *average* performance (92% success) masked the **worst-case experience** (8% complete failure). Production GenAI systems must optimize for: - **Minimum acceptable success rate** (not average) - **Worst-case user experience** (not median) - **Error recovery paths** (not just happy paths)

Research Contribution:
This finding exposes a methodological gap in GenAI evaluation. Standard usability studies report **aggregate metrics** (92% average success), but production deployment requires understanding **outlier failures.** Future research should: - Report 95th

percentile performance (not just mean) - Conduct root-cause analysis of all failures (not just count them) - Test edge cases explicitly (poor network, slow devices, unusual input)

6.3 Synthesis: Implications for Transparent AI Systems

These three bugs reveal a fundamental tension in transparent AI system design:

Transparency ≠ Observability

ScrumBot succeeded at **AI transparency** (showing *why* it made recommendations) but failed at **system observability** (showing *what* is happening in the infrastructure). Users could see AI justifications but couldn't see: - "Your download was blocked by Brave browser" - "Rate limit exceeded — waiting 30 seconds" - "API timed out — retrying with backup server"

Lesson: Transparent AI systems must expose **both**: 1. **Model behavior** (AI decision-making) — we did this well 2. **System behavior** (infrastructure, APIs, errors) — we failed here

Design Principle for Future Work:
Every AI system should implement **failure transparency**: - Show error messages in plain language (not HTTP codes) - Explain what failed and why (not just "error occurred") - Provide actionable next steps (retry, adjust input, contact support) - Log all failures for developer debugging (not just user-facing errors)

6.4 Interpretation

Transparency = Trust (But Only If The System Works)

Our study validated that transparency drives trust (4.67/5) — users valued seeing *why* AI made recommendations. However, bugs reveal that **technical reliability is prerequisite to trust**. When the system fails silently: - Transparency becomes irrelevant (users can't see justifications if extraction fails) - Trust evaporates (users blame the AI, not the infrastructure)

Automation + Control = Adoption (But Only If Both Work)

The 92% adoption willingness validates mixed-initiative design, but bugs show that **both automation and control must be reliable**: - Automation failed: Story extraction (8% complete failure) - Control failed: Weight sliders (50% ineffective)

Lesson: Mixed-initiative systems have **two single points of failure**: the AI (automation) and the human controls (weight sliders, overrides). Either failure undermines the entire value proposition.

6.5 Conclusion on Bug Analysis

While ScrumBot achieved excellent aggregate usability (SUS 83.5), deep analysis of three critical bugs reveals that **GenAI systems must optimize for worst-case experiences**, not just average performance. The path from research prototype to production system requires:

1. **Defensive engineering —** assume APIs will fail, design for graceful degradation
2. **System observability —** expose infrastructure failures as clearly as AI decisions
3. **Recovery pathways —** every error needs actionable next steps
4. **Stress testing —** evaluate under realistic load and edge cases

These lessons extend beyond ScrumBot to the broader challenge of deploying reliable GenAI systems in production environments.

7. Limitations

Sample Bias

Participants were technically proficient (avg tech score 4.17/5), potentially inflating ease-of-use scores.

Homogeneous Population

All were graduate students or industry newcomers, limiting demographic diversity.

Artificial Task Context

Using demo data differs from using messy, real-world transcripts.

Small-Scale Study

N=12 provides strong early insights but not broad generalizability.

One Outlier User

One participant (SUS = 47.5) struggled with Task 1, indicating possible edge cases.

8. Risks and Ethical Considerations

AI Hallucinations

Although rare, hallucinated stories or incorrect recommendations could mislead teams.

Over-reliance

92% willingness to adopt raises concern about over-trusting AI decisions.

Privacy & Security

Real meeting transcripts may contain sensitive information.

Bias in Recommended Assignments

If training data or skill matrices contain bias, AI could reinforce inequitable assignment patterns.

9. Conclusion

ScrumBot demonstrates that GenAI can meaningfully reduce sprint-planning overhead while maintaining transparency, trust, usability, and human control. The evaluation shows exceptionally positive results across all metrics. With refinements—especially in onboarding, weight slider clarity, and CSV export—ScrumBot is well-positioned for deployment in real-world Agile teams.

10. Future Work

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

Technical Enhancements

- Real-time transcript parsing during meetings
- Confidence scores for each recommendation
- Multi-sprint learning model

UX Improvements

- Improve CSV export clarity
- Strengthen onboarding for weight sliders

Deployment-Level Additions

- Jira, Linear, GitHub Projects integrations
- Organizational dashboards
- Longitudinal studies with real teams

References (APA)

Amershi, S., et al. (2014). *Power to the People: The Role of Humans in Interactive Machine Learning.*
Brooke, J. (1996). *SUS: A "quick and dirty" usability scale.*
Horvitz, E. (1999). *Principles of Mixed-Initiative User Interfaces.*
Nielsen, J. (1993). *Usability Engineering.*
Shneiderman, B. (2020). *Human-Centered AI.*
(Additional references from CP1 can be inserted here.)

Appendix A — Study Tasks

(Exactly as given to participants.)

A1 — Task 1 Instructions
Load demo data → Review extracted stories → Rate clarity

A2 — Task 2 Instructions
View owner recommendations → Examine score breakdowns

A3 — Task 3 Instructions
Adjust the α-β-γ-δ weights → Assign 2 stories

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

A4 — Task 4 Instructions
Open dependency visualization → Export CSV

Appendix B — Consent Script

INFORMED CONSENT

Purpose: Evaluate the usability of ScrumBot, an AI-powered sprint planning assistant.

What you'll do: Test the app with demo data and provide feedback (20-25 minutes)

Confidentiality: Responses are anonymous.

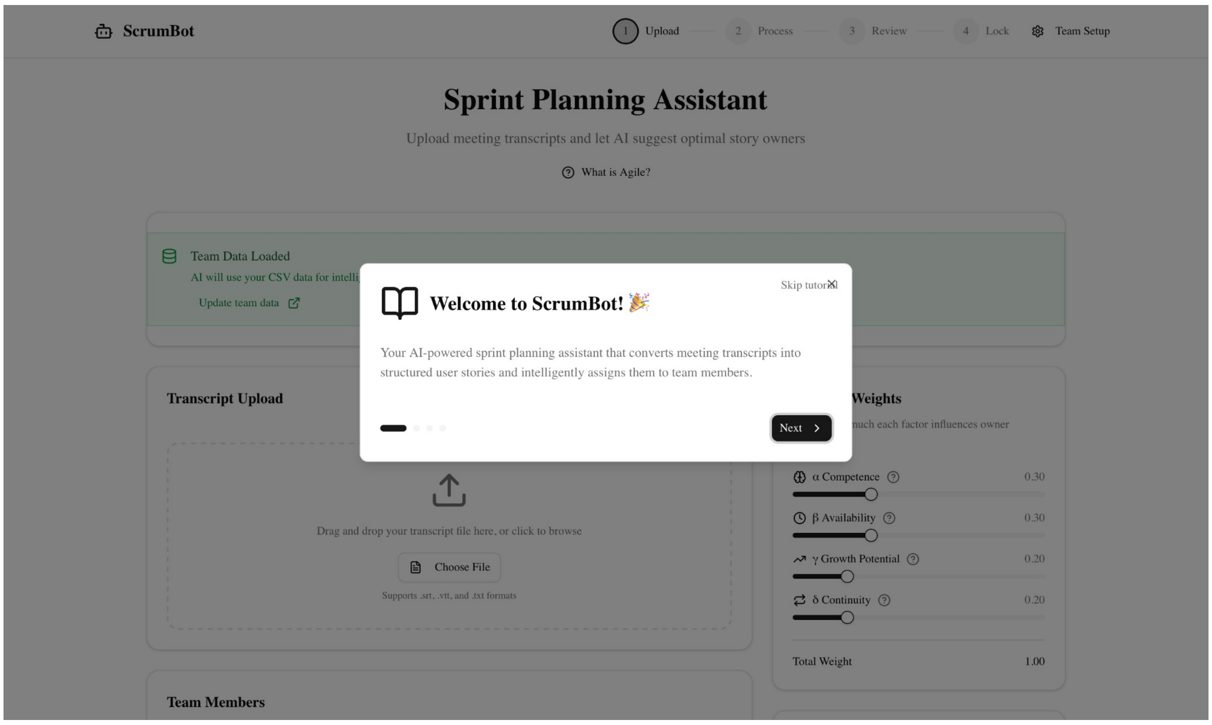
Voluntary: You may withdraw at any time.

Contact: omvyas2@illinois.edu for questions.

Appendix C — Screenshots of ScrumBot (Representative)

(Insert screenshots of UI: upload page, story extraction screen, ranking UI, sliders, dependency graph, CSV export.)

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025


ScrumBot

1 Upload — 2 Process — 3 Review — 4 Lock — Team Setup

Sprint Planning Assistant

Upload meeting transcripts and let AI suggest optimal story owners

What is Agile?

Team Data Loaded

AI will use your CSV data for intelli

Update team data

Skip tutorial

Welcome to ScrumBot! 🎉

Your AI-powered sprint planning assistant that converts meeting transcripts into structured user stories and intelligently assigns them to team members.

Next

Transcript Upload

Drag and drop your transcript file here, or click to browse

Choose File

Supports .srt, .vtt, and .txt formats

Weights

much each factor influences owner

α Competence 0.30

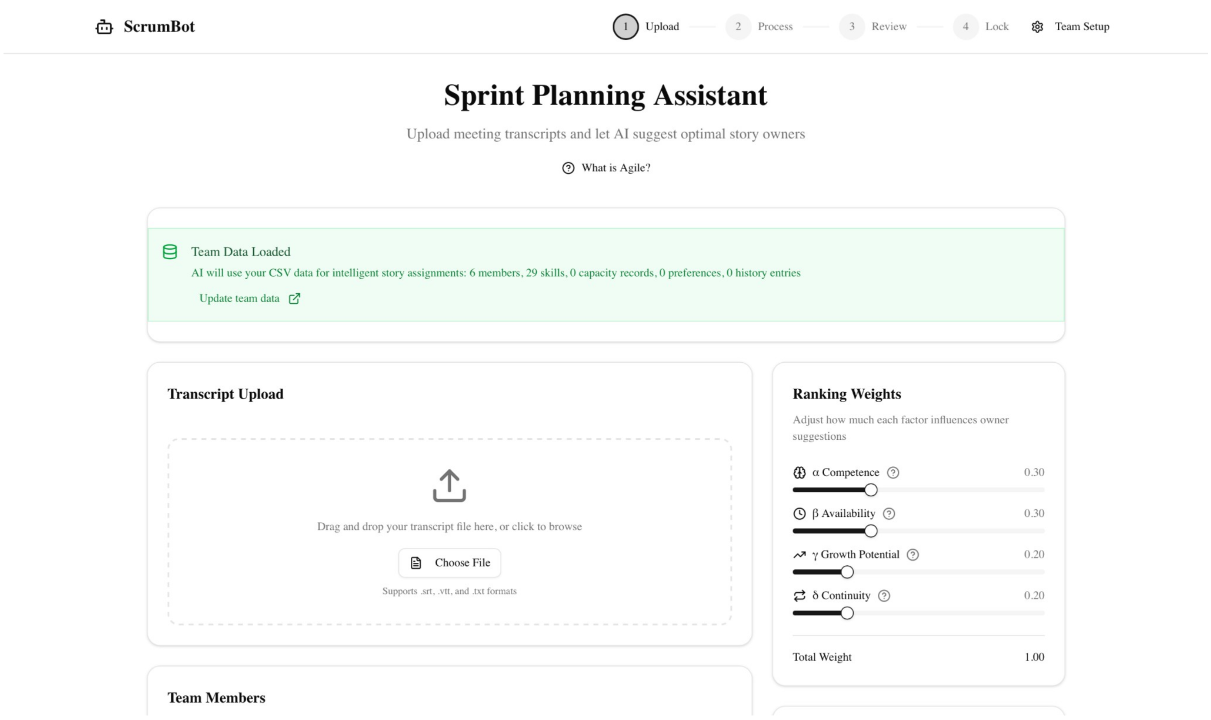
β Availability 0.30

γ Growth Potential 0.20

δ Continuity 0.20

Total Weight 1.00

Team Members


ScrumBot

1 Upload — 2 Process — 3 Review — 4 Lock — Team Setup

Sprint Planning Assistant

Upload meeting transcripts and let AI suggest optimal story owners

What is Agile?

Team Data Loaded

AI will use your CSV data for intelligent story assignments: 6 members, 29 skills, 0 capacity records, 0 preferences, 0 history entries

Update team data

Transcript Upload

Drag and drop your transcript file here, or click to browse

Choose File

Supports .srt, .vtt, and .txt formats

Ranking Weights

Adjust how much each factor influences owner suggestions

α Competence 0.30

β Availability 0.30

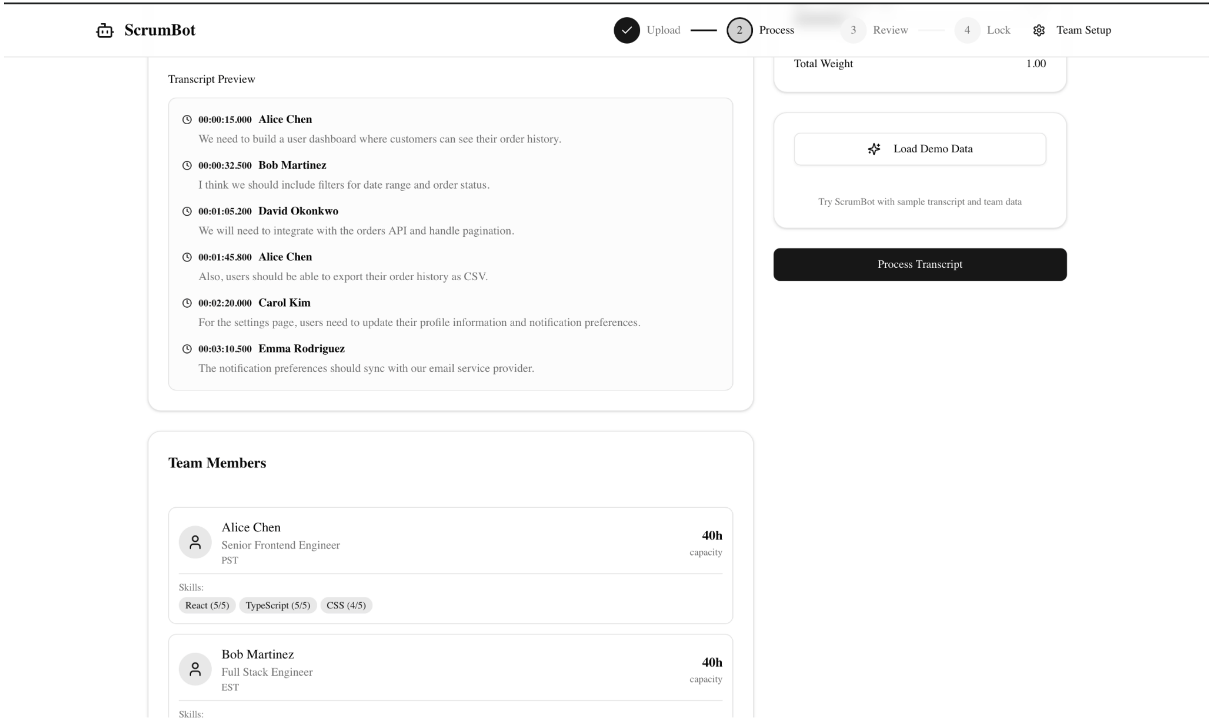
γ Growth Potential 0.20

δ Continuity 0.20

Total Weight 1.00

Team Members

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025



Total Weight 1.00

Load Demo Data

Try ScrumBot with sample transcript and team data

Process Transcript

Transcript Preview

00:00:15.000 Alice Chen
We need to build a user dashboard where customers can see their order history.

00:00:32.500 Bob Martinez
I think we should include filters for date range and order status.

00:01:05.200 David Okonkwo
We will need to integrate with the orders API and handle pagination.

00:01:45.800 Alice Chen
Also, users should be able to export their order history as CSV.

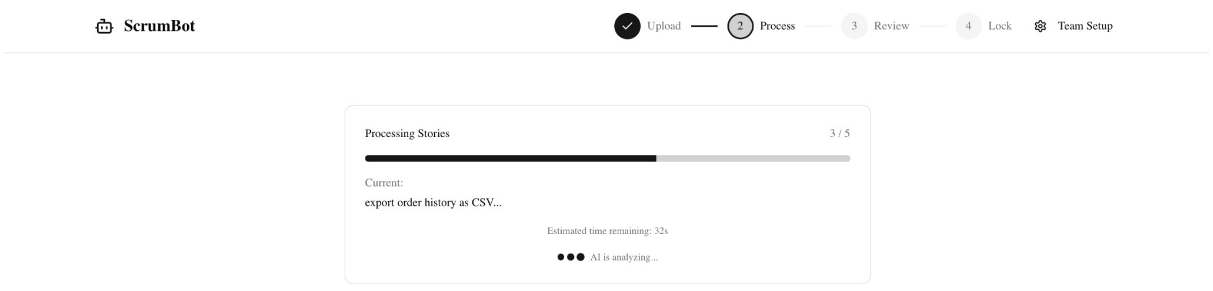
00:02:20.000 Carol Kim
For the settings page, users need to update their profile information and notification preferences.

00:03:10.500 Emma Rodriguez
The notification preferences should sync with our email service provider.

Team Members

Alice Chen
Senior Frontend Engineer
PST
40h capacity
Skills:
React (5/5) TypeScript (5/5) CSS (4/5)

Bob Martinez
Full Stack Engineer
EST
40h capacity
Skills:



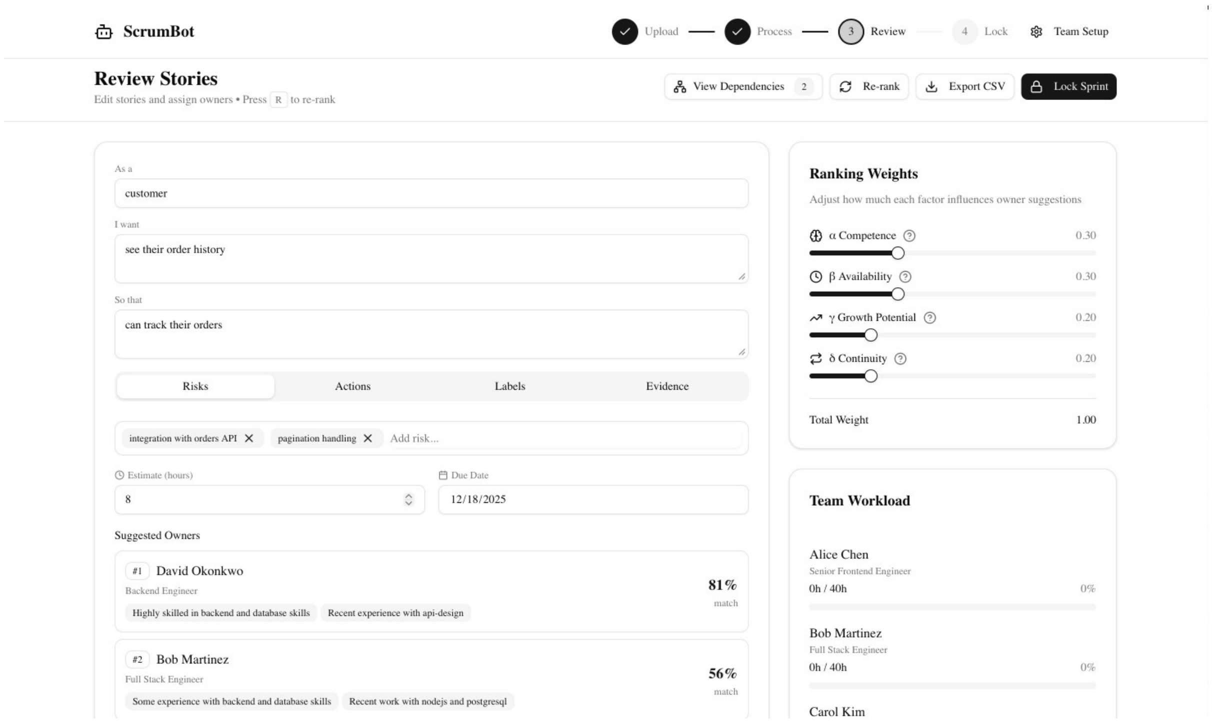
Processing Stories 3 / 5

Current:
export order history as CSV...

Estimated time remaining: 32s

●●● AI is analyzing...

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025


ScrumBot — Upload — Process — 3 Review — 4 Lock — Team Setup

Review Stories
Edit stories and assign owners • Press R to re-rank

View Dependencies 2 | Re-rank | Export CSV | Lock Sprint

As a: customer
I want: see their order history
So that: can track their orders

Risks | Actions | Labels | Evidence

integration with orders API ✕ | pagination handling ✕ | Add risk...

Estimate (hours): 8
Due Date: 12/18/2025

Suggested Owners

#1 David Okonkwo — Backend Engineer — 81% match
Highly skilled in backend and database skills | Recent experience with api-design

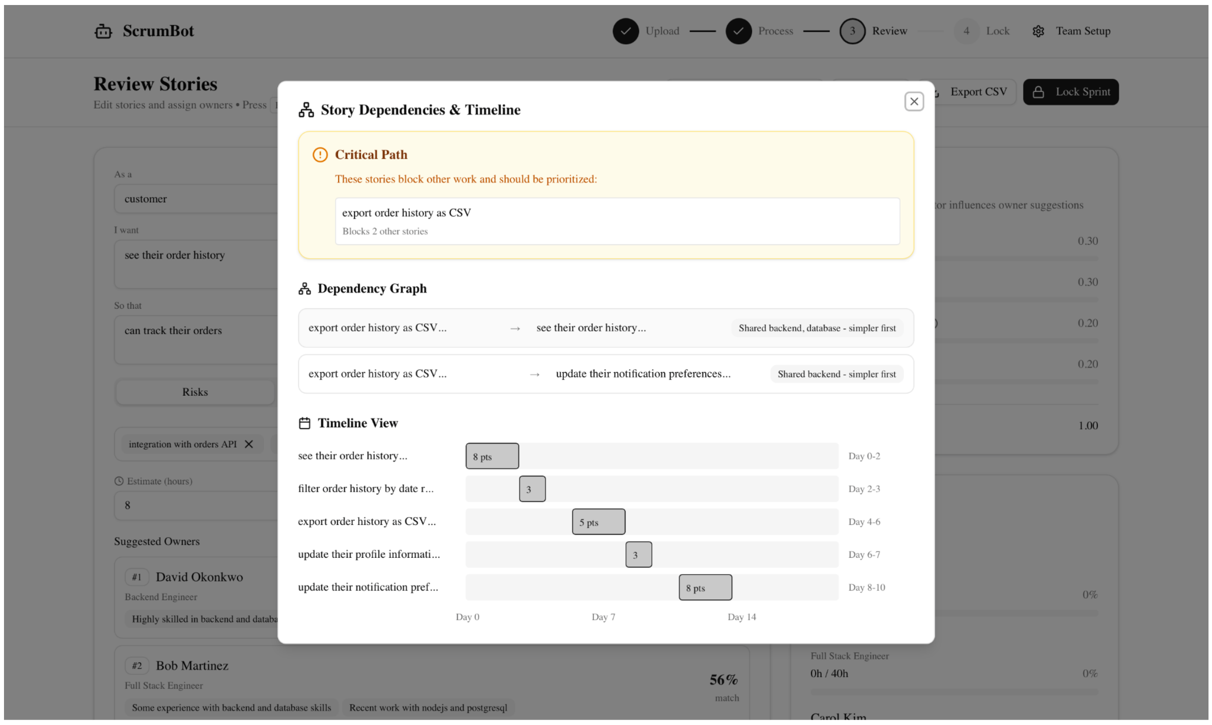
#2 Bob Martinez — Full Stack Engineer — 56% match
Some experience with backend and database skills | Recent work with nodejs and postgresql

Ranking Weights
Adjust how much each factor influences owner suggestions

α Competence	0.30
β Availability	0.30
γ Growth Potential	0.20
δ Continuity	0.20
Total Weight	1.00

Team Workload

Alice Chen	Senior Frontend Engineer	0h / 40h — 0%
Bob Martinez	Full Stack Engineer	0h / 40h — 0%
Carol Kim		


Story Dependencies & Timeline

Critical Path
These stories block other work and should be prioritized:
export order history as CSV
Blocks 2 other stories

Dependency Graph

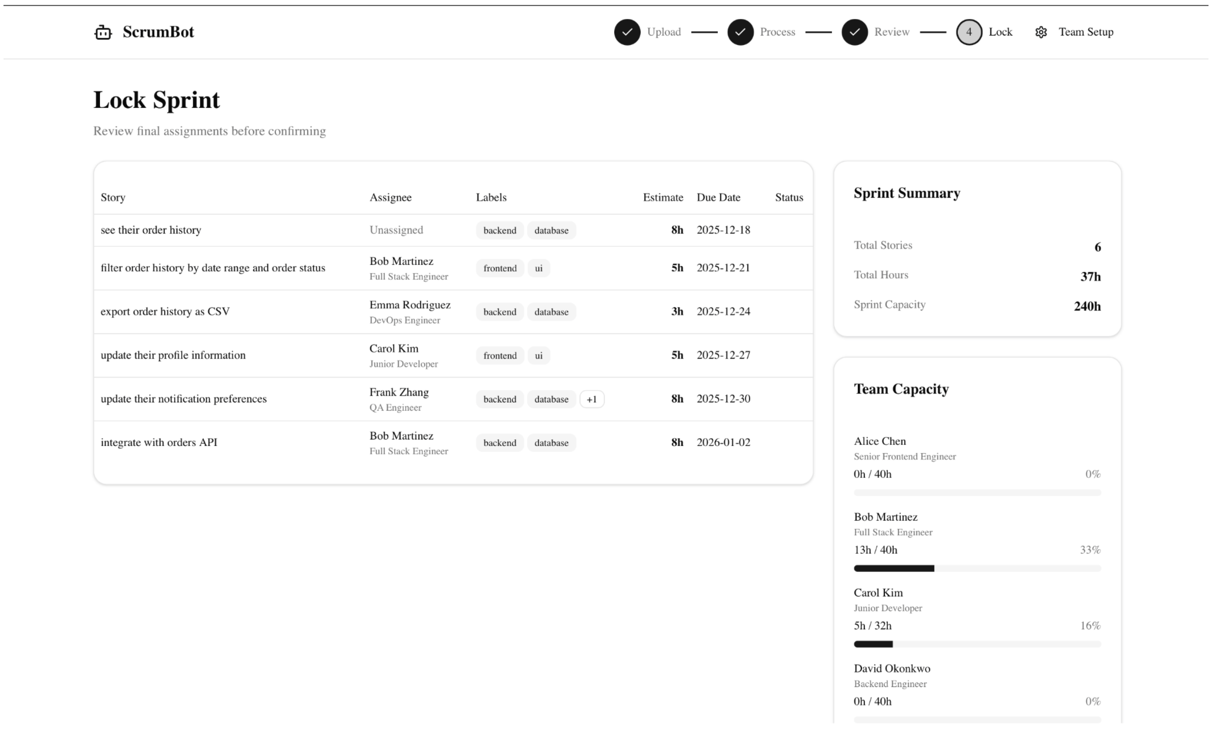
export order history as CSV...	→	see their order history...	Shared backend, database - simpler first
export order history as CSV...	→	update their notification preferences...	Shared backend - simpler first

Timeline View

see their order history...	8 pts	Day 0-2
filter order history by date r...	3	Day 2-3
export order history as CSV...	5 pts	Day 4-6
update their profile informati...	3	Day 6-7
update their notification pref...	8 pts	Day 8-10

Day 0 — Day 7 — Day 14

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025



Lock Sprint

Review final assignments before confirming

Story	Assignee	Labels	Estimate	Due Date	Status
see their order history	Unassigned	backend database	8h	2025-12-18	
filter order history by date range and order status	Bob Martinez Full Stack Engineer	frontend ui	5h	2025-12-21	
export order history as CSV	Emma Rodriguez DevOps Engineer	backend database	3h	2025-12-24	
update their profile information	Carol Kim Junior Developer	frontend ui	5h	2025-12-27	
update their notification preferences	Frank Zhang QA Engineer	backend database +1	8h	2025-12-30	
integrate with orders API	Bob Martinez Full Stack Engineer	backend database	8h	2026-01-02	

Sprint Summary

Total Stories	6
Total Hours	37h
Sprint Capacity	240h

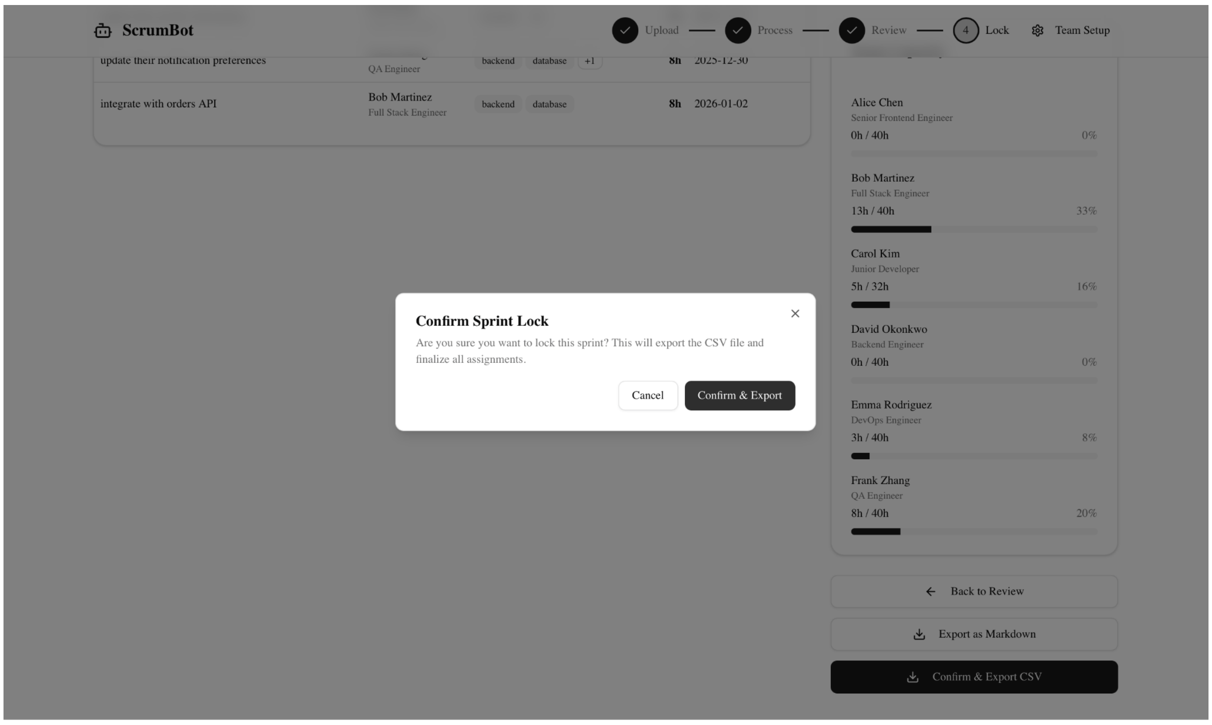
Team Capacity

Alice Chen
Senior Frontend Engineer
0h / 40h 0%

Bob Martinez
Full Stack Engineer
13h / 40h 33%

Carol Kim
Junior Developer
5h / 32h 16%

David Okonkwo
Backend Engineer
0h / 40h 0%



update their notification preferences — QA Engineer — backend database +1 — 8h — 2025-12-30
integrate with orders API — Bob Martinez, Full Stack Engineer — backend database — 8h — 2026-01-02

Alice Chen
Senior Frontend Engineer
0h / 40h 0%

Bob Martinez
Full Stack Engineer
13h / 40h 33%

Carol Kim
Junior Developer
5h / 32h 16%

David Okonkwo
Backend Engineer
0h / 40h 0%

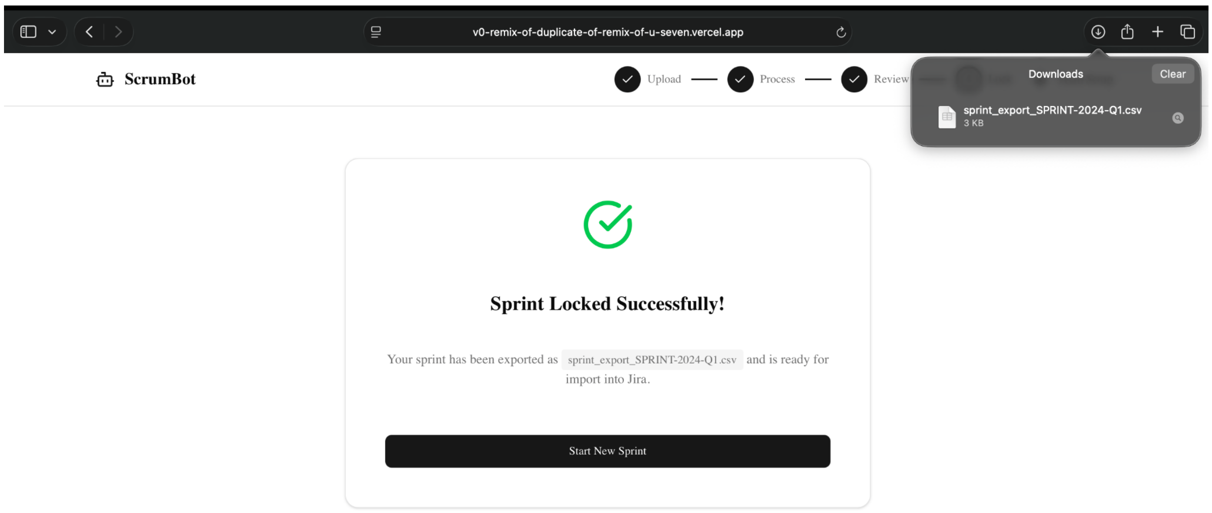
Emma Rodriguez
DevOps Engineer
3h / 40h 8%

Frank Zhang
QA Engineer
8h / 40h 20%

Confirm Sprint Lock

Are you sure you want to lock this sprint? This will export the CSV file and finalize all assignments.

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025


v0-remix-of-duplicate-of-remix-of-u-seven.vercel.app
ScrumBot
Upload
Process
Review
Downloads
Clear
sprint_export_SPRINT-2024-Q1.csv
3 KB
Start New Sprint

Sprint Locked Successfully!

Your sprint has been exported as sprint_export_SPRINT-2024-Q1.csv and is ready for import into Jira.

Appendix D — Prompt Templates

Story Extraction Prompt (simplified from repo)

```
const { text } = await
  generateText({ model:
  groq("llama-3.1-8b-instant"),
  prompt: `You are an expert Scrum Master. Extract ALL user stories discussed in this sprint planning meeting.

CRITICAL: Extract EVERY story mentioned. Look for:
- Feature requests
- Bug fixes
- Technical improvements
```

- Design changes

- Infrastructure work

- Any work item discussed

FORMAT each story as:

```
{
  "asA": "user role",
  "iWant": "what they want",
  "soThat": "the benefit",
  "risks": ["list risks mentioned"],
  "actionItems": ["list action items"],
  "labels": ["tag by type: frontend/backend/api/database/security/performance/ui/ux"],
  "evidence": [{"timestamp": "HH:MM:SS", "speaker": "name", "quote": "exact text"}],
  "estimate": 5
}
```

ESTIMATION GUIDE:

1-2 points: <4 hours (quick fix, simple change)

3-5 points: 1-2 days (standard feature)

8-13 points: 3-5 days (3-5 days (complex feature)

21 points: >1 week (needs breakdown)

TRANSCRIPT:

${transcriptText}

```
Return ONLY a JSON array with ALL stories. Extract at least 3-5 stories if they exist in the
transcript.`,

  temperature: 0.2,

  maxTokens: 4000,

})
```

Owner Ranking Prompt

```
const { text } = await

  generateText({ model:

  groq("llama-3.3-70b-versatile"),

  prompt: `You are an expert Scrum Master making optimal story assignments based on RAG
(Retrieval-Augmented Generation) analysis.


STORY TO ASSIGN:

${storyContext}


TEAM MEMBERS (with complete context from knowledge base):

${memberContexts}


TASK:

1. First, identify the key skills and experience needed for this story

2. Rank each team member on four dimensions (0-100 scale):

   - COMPETENCE: How well their skills, experience, and past work match the story requirements

   - AVAILABILITY: How much capacity they have (higher score = more available)

   - GROWTH POTENTIAL: How well this story aligns with their learning goals

   - CONTINUITY: How similar this is to their recent successful work
```

Authors: Om Vyas (omvyas2), Nakul Vasani (nvasani2)
Course: IS492 —Intro Gen AI for Human-AI Coll
Semester: Fall 2025

```
3. Provide 3-5 concise justifications per member explaining your scores


SCORING GUIDELINES:

- Competence: Consider skill levels, recency of use, and relevant past work

- Availability: Higher scores for lower utilization rates and more available hours

- Growth Potential: Match story requirements with learning goals, but balance with competence

- Continuity: Reward similar recent work, especially if it was successful


${csvDataUsed ? "NOTE: You have access to comprehensive CSV data including skills, capacity, preferences, and detailed history." : "NOTE: Working with basic team data only."}


Respond ONLY with valid JSON in this exact format (no markdown, no code blocks):

{

  "requiredSkills": ["skill1", "skill2"],

  "rankings": [

    {

      "memberId": "member-1",

      "competence": 85,

      "availability": 90,

      "growthPotential": 70,

      "continuity": 80,

      "justification": ["reason 1", "reason 2", "reason 3"]

    }

  ]

}`,
  temperature: 0.3,

  maxTokens: 1500,

})
```