

Sprint Visibility & Code Intelligence Platform

Core Problem

Technical leaders and CTOs lack real-time visibility into their team's development work. They struggle to:

- Understand who's working on what features each sprint
- See how developers are solving problems without diving into code
- Identify team members' contributions and effort
- Provide timely, relevant feedback and suggestions
- Communicate with non-technical stakeholders about progress

Solution

An AI-powered dashboard that translates Git activity into plain-language insights, giving leaders sprint-level visibility without requiring them to read code.

Key Features

1. Smart Sprint Dashboard

- Real-time view of all active features and their developers
- Auto-generated summaries: "Sarah is building user authentication using OAuth 2.0"
- Visual contribution metrics per developer

2. AI Code Translator

- Converts commits into business-readable updates
- Example: "Fixed payment gateway timeout" instead of "Refactored async handlers in stripe.service.js"
- Shows problem-solving approach in simple terms

3. Effort Recognition Engine

- Tracks meaningful contributions beyond line counts
- Identifies code quality, complexity tackled, blockers resolved
- Highlights who's going above and beyond

4. Contextual Feedback Loop

- Leaders can comment directly on features/commits
- Suggestions auto-route to relevant developers
- Non-technical stakeholders can ask questions in plain language

5. Sprint Intelligence Reports

- Weekly summaries of feature progress
- Risk flags (stalled work, merge conflicts, blockers)
- Team velocity and collaboration patterns

User Flow Example

Monday Morning:

CTO opens dashboard → sees "3 features in progress, 2 ready for review"

Clicks on 'Checkout Optimization':

- **Developer:** John Smith
- **What he's doing:** "Reducing cart abandonment by adding one-click guest checkout"
- **How he's solving it:** "Storing temporary session data and simplifying the payment form"
- **Progress:** 60% complete, 12 commits this week
- **Effort level:** High (tackling complex state management)

CTO adds suggestion: "Consider mobile wallet integration for faster checkout" → John gets notified with context

Target Users

Primary:

- Engineering Managers (10-50 person teams)
- CTOs at growth-stage companies
- Technical Product Managers

Secondary:

- Non-technical founders/stakeholders wanting dev transparency
- Remote team leaders needing async visibility

Differentiation

vs. GitHub/GitLab: We translate code into business outcomes, they show raw commits

vs. Jira/Linear: We show *how* work is done from actual code, they just track tasks

vs. Code analytics tools: We focus on human-readable insights for leaders, not developer metrics

Implementation Considerations

1. **Integrations needed:** GitHub, GitLab, Bitbucket, Jira, Linear
2. **AI requirements:** Code parsing + LLM for plain-language translation
3. **Privacy:** Opt-in for teams, no code sharing outside org
4. **Pricing model:** Per-user SaaS or team-based tiers

Validation Steps

1. Interview 20 CTOs/engineering managers about current visibility pain points
2. Build MVP with GitHub integration only
3. Test AI translation accuracy with real commits
4. Pilot with 3-5 teams for one sprint cycle

Potential Name Ideas

- **CodePulse** - Real-time heartbeat of your development
- **SprintLens** - Clear view into sprint progress
- **DevInsight** - Developer work, manager clarity
- **Commit Clarity** - From code to conversation

Next Steps

1. Which pain point resonates most with your experience?
2. Would you start with engineering managers or CTOs?
3. Should the MVP focus on visibility or feedback features first?