 # AI Agent Assignment: Team Activity Monitor (2-Day Sprint)
Project Overview

Build a simple AI chatbot that integrates with JIRA and GitHub APIs to answer basic questions about team member activities. This is a rapid prototyping exercise to demonstrate API integration and problem-solving skills.

Objective
Create a working prototype that can answer the question: **"What is [member] working on these days?"** by fetching data from JIRA and GitHub.

Core Requirements
1. Simple Chat Interface
    Basic web interface with input/output (can be simple HTML + JavaScript)
    OR command-line interface
    Accept natural language questions about team members
2. JIRA Integration
Required endpoints:
    Get assigned issues for a user
    Fetch issue status and recent updates
    Basic authentication (API token)
Example queries to handle:*
    "What JIRA tickets is John working on?"
    "Show me Sarah's current issues"
3. GitHub Integration
Required endpoints:
    Get recent commits by user
    Fetch active pull requests
    List repositories user contributed to recently
Example queries to handle:
    "What has Mike committed this week?"
    "Show me Lisa's recent pull requests"
4. AI Response Generation
    Use OpenAI API, Claude API, or simple template-based responses
    Combine JIRA and GitHub data into human-readable answers
    Handle basic error cases (user not found, no recent activity)
Technical Requirements
Minimal Tech Stack
    Backend: Node.js/Python (Express/Flask)
    Frontend: Simple HTML/CSS/JS or CLI

AI: OpenAI API (GPT-3.5) or template responses
APIs: JIRA REST API + GitHub REST API

Must-Have Features

● Authentication: Basic API token auth for JIRA and GitHub ● User Query Processing: Parse user questions to extract member names ● Data Fetching: Get recent activity from both platforms
● Response Formatting: Present data in conversational format ##

Implementation Tasks

Core Development Tasks

[ ] Project setup and environment configuration
[ ] JIRA API authentication and basic connection
[ ] Implement endpoint to fetch user's assigned issues
[ ] GitHub API authentication and basic connection
[ ] Implement endpoint to fetch user's recent commits and PRs
[ ] Create simple data processing functions
[ ] Test both API integrations independently
[ ] Implement basic query parsing (extract user names from questions) [ ] Integrate AI API for response generation OR create response templates [ ] Combine JIRA and GitHub data into coherent answers
[ ] Handle basic error scenarios
[ ] Build simple user interface (web or CLI)
[ ] End-to-end testing with real data
[ ] Documentation and demo preparation
[ ] Code cleanup and final testing

Deliverables

Required (End of Day 2):

● Working Application**: Functional chatbot that answers the core question ● Source Code**: Clean, commented code with clear structure ● **Demo: 10-minute demonstration of the working system
● Basic Documentation**: Setup instructions and API usage

Test Cases to Implement:

"What is John working on these days?"
"Show me recent activity for Sarah"
"What has Mike been working on this week?"
Handle case when user has no recent activity
Handle case when user is not found

Sample Implementation Structure

```
project/
```

```
├── src/
│   ├── jira-client.js # JIRA API integration
│   ├── github-client.js # GitHub API integration
│   ├── query-parser.js # Extract user names from queries
│   ├── response-generator.js # Format responses
│   └── main.js # Main application logic
├── public/
│   ├── index.html # Simple web interface
│   └── script.js # Frontend logic
├── config/
│   └── config.js # API keys and configuration
└── README.md # Setup and usage instructions
```

Evaluation Criteria

Technical Implementation (50%)

- Working API integrations with error handling
- Clean, readable code structure
- Proper configuration management
- Basic security practices (no hardcoded secrets)

Functionality (30%)

- Successfully answers core questions
- Handles basic error cases
- Presents data in readable format
- Demonstrates understanding of both APIs

Problem-Solving & Efficiency (20%)

- Efficient use of 2-day timeline
- Creative solutions to technical challenges
- Good use of available resources and documentation
- Clear communication of technical decisions

Provided Resources

API Access:

- JIRA instance URL and API token (will be provided)
- GitHub personal access token with appropriate permissions
- Sample user data for testing

Documentation Links:

- [JIRA REST API Quick Start](https://developer.atlassian.com/server/jira/platform/rest-apis/)

[GitHub REST API Basics](https://docs.github.com/en/rest/quickstart) [OpenAI API Quick Start](https://platform.openai.com/docs/quickstart) ## Success Criteria

Minimum Viable Product:

- ✅ User can ask “What is [name] working on?”
- ✅ System fetches data from both JIRA and GitHub
- ✅ Provides a readable response combining both sources
- ✅ Handles at least one error case gracefully

Bonus Points:

- Multiple question formats supported
- Nice user interface design
- Additional insights (time estimates, priority levels)
- Performance optimizations (caching, concurrent requests)

Final Demo Format (15 minutes)

- Quick walkthrough** of the code structure (3 min)
- Live demonstration** with sample queries (7 min)
- Technical challenges** faced and solutions (3 min)
- Q&A** and potential improvements (2 min)

Timeline**: 2 working days
Goal**: Demonstrate rapid prototyping skills and API integration capabilities
Focus**: Working solution over perfect code - prioritize functionality and clear communication