**StoryWeaver Agent Documentation**

**Overview**

StoryWeaver is an intelligent agent built using LangGraph that helps product owners manage, rate, and refine user stories based on INVEST principles (Independent, Negotiable, Valuable, Estimable, Small, Testable). The agent integrates with Jira to retrieve, analyze, and update user stories, providing a conversational interface for efficient user story management and refinement.

**Architecture Components**

**User Interface Layer**

* **Gradio Web App**: A browser-based chat interface for interacting with the StoryWeaver agent
* **PyQt5 Desktop App**: A native desktop application offering the same functionality with offline capabilities
* **Configuration Interface**: Both apps include screens for setting up Jira credentials and preferences

**Agent Layer**

* **LangGraph Agent**: Core component implementing the agent's reasoning and tool orchestration
* **Configuration Storage**: Local configuration file storing Jira credentials and user preferences
* **Chat Completion API**: External API for processing natural language and generating responses

**Integration Layer**

* **Jira API Integration**: Direct connection to Jira to retrieve and update issues

**Agent Tools and Capabilities**

**1. Get All Issues**

* **Function**: Retrieves all user stories from Jira based on status (e.g., "To Do", "In Progress", "Done")
* **Input**: User status filter
* **Output**: List of user stories with ID, title, and brief description

**2. Get Issue By ID**

* **Function**: Retrieves detailed information about a specific user story
* **Input**: Issue ID or reference (e.g., "3rd one")
* **Output**: Complete user story details including description, acceptance criteria, and current status

**3. Rate Story**

* **Function**: Evaluates a user story against INVEST principles
* **Input**: User story details
* **Output**: Rating for each INVEST principle with explanations and an overall score

**4. Refine Story**

* **Function**: Suggests improvements to make the story better aligned with INVEST principles
* **Input**: User story and its INVEST rating
* **Output**: Refined user story with specific improvements for each principle that scored low

**5. Update Jira**

* **Function**: Pushes the refined user story back to Jira
* **Input**: Refined user story content
* **Output**: Confirmation of update with link to updated issue

**Setup and Configuration**

**Initial Configuration**

1. Launch either the Gradio web app or PyQt5 desktop application
2. Navigate to the configuration screen
3. Provide the following information:
   * Jira URL (e.g., https://your-organization.atlassian.net)
   * Jira API access token
   * Jira user ID (email associated with your Jira account)
4. Optional: Configure use case details for improved agent performance
5. Save configuration to local config file

**User Interaction Flow**

**1. List Stories**

* **User Command**: "list my stories"
* **Agent Action**: Calls the "Get All Issues" tool
* **Result**: Displays a numbered list of stories from Jira

**2. Select a Story**

* **User Command**: "pick the 3rd one"
* **Agent Action**: Calls the "Get Issue By ID" tool to retrieve details of the third story
* **Result**: Displays the complete details of the selected story

**3. Rate the Story**

* **User Command**: "rate it"
* **Agent Action**: Uses the "Rate Story" tool to evaluate against INVEST principles
* **Result**: Provides a detailed rating with scores and explanations for each principle

**4. Refine the Story**

* **User Command**: "refine it"
* **Agent Action**: Uses the "Refine Story" tool to generate improvements
* **Result**: Presents a refined version of the story with specific enhancements

**5. Update in Jira**

* **User Command**: "update the jira"
* **Agent Action**: Uses the "Update Jira" tool to push changes
* **Result**: Confirms the update was successful and provides a link to the updated issue

**INVEST Principles Evaluation Criteria**

Each user story is evaluated against the following criteria:

**Independent**

* Does the story stand alone without dependencies?
* Can it be developed and delivered independently?

**Negotiable**

* Is there room for discussion on implementation details?
* Is it a requirement rather than a specific solution?

**Valuable**

* Does it provide clear value to users or stakeholders?
* Is the value explicitly stated or easily inferred?

**Estimable**

* Is there enough information to estimate the effort?
* Are the requirements clear enough for planning?

**Small**

* Is the scope limited enough to complete in a single sprint?
* Can it be broken down further if needed?

**Testable**

* Are the acceptance criteria clear and testable?
* Can the team determine when it's "done"?

**Agent Workflow**

1. **Input Processing**:
   * The agent receives natural language input from the user
   * The input is processed through the Chat Completion API
   * The agent determines the user's intent
2. **Tool Selection**:
   * Based on the intent, the agent selects the appropriate tool
   * The agent may request additional information if needed
3. **Tool Execution**:
   * The selected tool is executed with the necessary parameters
   * For Jira operations, the agent uses the stored credentials
4. **Response Generation**:
   * The agent formats the tool output into a natural language response
   * The response is sent back to the user through the chat interface
5. **Conversation Context**:
   * The agent maintains context throughout the conversation
   * Previous selections and actions influence subsequent interactions

**Technical Implementation Details**

**LangGraph Implementation**

The StoryWeaver agent is implemented using LangGraph, which provides:

* A graph-based workflow for complex reasoning
* State management for conversational context
* Tool integration framework for external API calls

**User Interface Options**

* **Gradio Web App**:
  + Browser-based interface
  + Simple deployment
  + Accessible from any device with a web browser
  + Responsive design for mobile accessibility
* **PyQt5 Desktop App**:
  + Native application for Windows, macOS, and Linux
  + Enhanced performance and responsiveness
  + System-level integrations

**Integration Patterns**

* **Jira REST API**: RESTful API calls with OAuth authentication
* **Synchronous Communication**: Direct API calls for real-time interaction
* **Stateful Sessions**: Conversation state maintained throughout user sessions

**Error Handling and Resilience**

* **Authentication Failures**: Prompts for re-authentication or configuration update
* **Invalid Commands**: Provides helpful guidance when user commands are unclear
* **Tool Failures**: Graceful error handling with descriptive error messages

**Best Practices for Product Owners**

1. Be specific with story selection commands
2. Review ratings before requesting refinements
3. Verify refined stories before updating Jira
4. Use the agent iteratively to improve stories incrementally
5. Provide feedback on refinements that don't align with team goals
6. Use the agent during backlog grooming sessions to improve story quality

**Benefits for Product Owners**

1. **Quality Improvement**: Consistent evaluation of user stories against industry-standard principles
2. **Time Savings**: Automates the refinement process, focusing product owner attention where it matters
3. **Team Alignment**: Creates a shared understanding of what makes a good user story
4. **Backlog Health**: Systematically improves the overall quality of the product backlog
5. **Consistency**: Applies the same evaluation criteria across all stories

**Limitations**

* The agent evaluates stories based on INVEST principles but cannot account for all organizational context
* Complex dependencies between stories may not be fully captured
* The refinement suggestions are based on best practices and may need adaptation for specific teams
* The agent works best with well-structured Jira projects

**Example Conversation**

Product Owner: list my stories

Agent: I've found the following user stories in your Jira:

1. ABC-123: "As a user, I want to reset my password"

2. ABC-124: "As an admin, I need to view user activity logs"

3. ABC-125: "Users should be able to export their data"

4. ABC-126: "The system should send email notifications"

Product Owner: pick the 3rd one

Agent: I've retrieved details for story ABC-125:

Title: Users should be able to export their data

Description: Users need a way to export their personal data in a common format

Acceptance Criteria:

- Data should be exportable in CSV and JSON

- Export should include all user-generated content

- Process should complete within 30 seconds

Status: To Do

Priority: Medium

Product Owner: rate it

Agent: I've rated story ABC-125 based on INVEST principles:

Independent: 7/10 - Mostly independent but may have some UI dependencies

Negotiable: 5/10 - Somewhat prescriptive about formats (CSV and JSON)

Valuable: 8/10 - Clear user value in data portability

Estimable: 6/10 - Mostly estimable but "all user-generated content" is ambiguous

Small: 7/10 - Reasonably scoped for a sprint

Testable: 8/10 - Good acceptance criteria, though "all content" needs clarification

Overall Score: 6.8/10

Product Owner: refine it

Agent: Here's a refined version of the story:

Title: "As a user, I want to export my personal data so that I can keep a backup or transfer it elsewhere"

Description: Users need a way to export their personal profile information and content in standard formats.

Acceptance Criteria:

- Users can select which data categories to export (profile info, posts, comments)

- Data is exportable in CSV and JSON formats

- Export process provides progress indicator and completes within 45 seconds for accounts with standard usage

- Exported data includes timestamps and metadata

- Users receive a notification when their export is ready

This refinement improves:

- Independence: Clearer scope boundaries

- Negotiability: More focus on the why, less on implementation

- Value: Explicitly states the benefit

- Estimability: More specific about data scope

- Testability: More detailed acceptance criteria

Product Owner: update the jira

Agent: I've updated the story in Jira. Story ABC-125 has been successfully updated with the refined content. You can view it here: https://your-organization.atlassian.net/browse/ABC-125