Product Requirements Document: Jira AI Agent

1. Overview

1.1 Product Description

The Jira AI Agent is an intelligent assistant that leverages Google Cloud's Agent Development Kit (ADK) to automate the creation and management of Jira tasks. It analyzes documents from Google Drive to identify potential tasks, verifies if similar tasks already exist in Jira, and facilitates the creation of new tasks with user feedback.

1.2 Objectives

- Automate the identification of potential tasks from unstructured documents
- Reduce manual effort in Jira task creation and management
- Prevent duplicate tasks through intelligent similarity checking
- Provide a user-friendly interface for reviewing and approving suggested tasks
- Streamline the process of moving work items from documentation to actionable Jira tasks

1.3 Target Users

- Project managers
- Product owners
- Business analysts
- Quality assurance teams

2. Technical Architecture

2.1 Core Technologies

- **Google Cloud Agent Development Kit (ADK)**: Foundation for building the AI agent
- **Google Cloud Multimodal Content Processing (MCP) Tools**: For processing various content types
- **Jira MCP**: For task creation and management
- **Google Drive API**: For accessing source documents

2.2 Jira MCP Integration

For the Jira integration, we will build a custom Jira MCP tool since there isn't a pre-built one available in the Google Cloud ecosystem. If you want to use a MCP that is available on github for Jira Cloud, please confirm with us which one you intend to use and why. This custom MCP will:

- Connect securely to Jira instances
- Query existing issues
- Create new issues
- Update issue statuses
- Handle authentication and authorization

2.3 System Architecture

```
[Google Drive] → [ADK Agent] → [Jira MCP] → [Jira Cloud/Server]
                    ↑
   [User Interface] ↔ [Issue Suggestion Engine]
```

3. Functional Requirements

3.1 Document Processing

- **FR1.1**: The agent shall integrate with Google Drive to access user-specified documents
- **FR1.2**: The agent shall support common document formats (PDF, DOCX, TXT, Google Docs, MarkDown)
- **FR1.3**: The agent shall parse document content to extract potential actionable items
- **FR1.4**: The agent shall maintain document context when suggesting tasks

3.2 Task Identification

- **FR2.1**: The agent shall identify potential tasks from parsed documents
- **FR2.2**: The agent shall classify identified tasks by type (bug, feature, improvement, etc.)
- **FR2.3**: The agent shall assign appropriate priority levels to suggested tasks
- **FR2.4**: The agent shall extract relevant metadata for each suggested task (components, labels)
- **FR2.5**: The agent shall suggest appropriate assignees based on task content and project history

3.3 Duplicate Detection

- **FR3.1**: The agent shall query existing Jira issues to identify potential duplicates
- **FR3.2**: The agent shall use semantic similarity to match suggested tasks with

existing issues
- **FR3.3**: The agent shall provide confidence scores for potential duplicates
- **FR3.4**: The agent shall present potential duplicates alongside suggested new tasks

3.4 User Interface

- **FR4.1**: The agent shall provide a web interface for reviewing suggested tasks
- **FR4.2**: The agent shall display task details including title, description, priority, and type
- **FR4.3**: The agent shall allow users to edit suggested task details before creation
- **FR4.4**: The agent shall provide batch approval/rejection capabilities
- **FR4.5**: The agent shall show source document context for each suggested task

3.5 Task Creation

- **FR5.1**: The agent shall create tasks in Jira based on user-approved suggestions
- **FR5.2**: The agent shall support custom fields defined in the Jira project
- **FR5.3**: The agent shall provide status updates during batch task creation
- **FR5.4**: The agent shall handle errors during task creation gracefully

4. Non-Functional Requirements

4.1 Performance

- **NFR1.1**: The agent shall process documents up to 50 pages within 2 minutes
- **NFR1.2**: The agent shall support concurrent processing of up to 10 documents
- **NFR1.3**: The agent shall complete duplicate detection within 30 seconds per suggested task
- **NFR1.4**: The agent shall create Jira tasks within 5 seconds of user approval

4.2 Security

- **NFR2.1**: The agent shall authenticate users through Google Cloud IAM
- **NFR2.2**: The agent shall store Jira credentials securely using Google Cloud Secret Manager
- **NFR2.3**: The agent shall maintain audit logs of all task creation activities
- **NFR2.4**: The agent shall enforce role-based access controls for task creation approval

4.3 Scalability

- **NFR4.1**: The agent shall support up to 30 concurrent users
- **NFR4.2**: The agent shall handle projects with up to 10,000 existing Jira issues
- **NFR4.3**: The agent shall scale resources automatically based on usage patterns

5. User Experience Flow

5.1 Document Selection

1. User logs into the AI Agent web interface
2. User selects a document from connected Google Drive
3. User initiates document processing

5.2 Task Review

1. Agent presents suggested tasks extracted from the document
2. User reviews each suggested task with contextual information
3. User sees potential duplicate issues from Jira
4. User can modify task details (title, description, type, priority, etc.)
5. User selects tasks for creation

5.3 Task Creation

1. User approves selected tasks for creation
2. Agent creates tasks in Jira with specified details
3. Agent provides confirmation with links to created tasks
4. User can view task creation history and status

6. Integration Requirements

6.1 Google Cloud ADK Integration

- **IR1.1**: The agent shall be built using the Google Cloud Agent Development Kit
- **IR1.2**: The agent shall leverage ADK's natural language understanding capabilities
- **IR1.3**: The agent shall use ADK's conversation management for user interactions

6.2 Google Cloud MCP Tools Integration

- **IR2.1**: The agent shall use the MCP tools for document processing
- **IR2.2**: The agent shall extract structured information from unstructured content

- **IR2.3**: The agent shall maintain document context during content processing

6.3 Custom Jira MCP Integration

- **IR3.1**: The agent shall build a custom Jira MCP for bidirectional communication with Jira
- **IR3.2**: The MCP shall support various Jira deployment types (Cloud, Server, Data Center)
- **IR3.3**: The MCP shall implement Jira REST API best practices
- **IR3.4**: The MCP shall handle authentication via API tokens and OAuth

6.4 Google Drive Integration

- **IR4.1**: The agent shall connect to Google Drive via the Google API
- **IR4.2**: The agent shall request the minimal permissions required for document access
- **IR4.3**: The agent shall support shared documents and team drives

7. Deployment Requirements

7.1 Infrastructure

- **DR1.1**: The agent shall be deployed on Google Cloud Platform
- **DR1.2**: The agent shall use container-based deployment for scalability
- **DR1.3**: The agent shall leverage Google Cloud Run for serverless operation

7.2 References

- Google Cloud Agent Development Kit Documentation
- Google Cloud MCP Tools Documentation
- ADK Tools Documentation
- Jira REST API Documentation

Jira AI Agent: Resource Plan & Hour Estimate

1. Introduction

This document outlines the estimated resource allocation and effort required to develop the Jira AI Agent as described in the Product Requirements Document (PRD). The plan breaks down the project into logical phases, identifies the necessary roles, and estimates hours for each major task area.

2. Required Roles

The following roles are anticipated for the successful execution of this project:

- **Project Manager (PM):** Oversees project execution, manages timelines, resources, and communication.
- **AI/ML Engineer:** Designs and implements the AI components, including document parsing, task identification, classification, and duplicate detection logic using ADK and MCP tools.
- **Backend Developer:** Develops the core application logic, integrations (Google Drive), the custom Jira MCP, API handling, and database interactions. Implements security measures.
- **Frontend Developer:** Develops the user interface for task review, modification, and approval.
- **DevOps Engineer:** Sets up and manages the cloud infrastructure (GCP), deployment pipelines (CI/CD), containerization, monitoring, and ensures scalability and security configurations.
- **QA Engineer:** Develops and executes test plans, performs functional, performance, security, and integration testing.

3. Resource Allocation & Estimated Hours per Phase

Phase / Major Task Area	**Primary Role(s)**	**Estimated Hours**	**Key Activities & Requirements Covered**
Phase 1: Planning & Setup	PM, AI/ML Eng, Backend Dev, DevOps	**40 hrs**	Project kickoff, detailed planning, environment setup (GCP, ADK), Jira API familiarization, Custom MCP initial design.
Phase 2: Core Backend & AI Dev	AI/ML Eng, Backend Dev	**300 hrs**	Google Drive Integration (FR1.1, IR4), Document Processing (FR1.2-1.4, IR2), Task Identification/Classification (FR2, IR1.2), Duplicate Detection

			(FR3), Custom Jira MCP Build (2.2, IR3).
Phase 3: Frontend Development	Frontend Dev	**80 hrs**	UI Development (FR4), Task review/edit/approval flow, Context display, Batch actions, Integration with backend APIs.
Phase 4: Integration & Security	Backend Dev, DevOps, All Devs	**45 hrs**	Integrating all components (ADK, MCPs, UI, Backend), Security Implementation (NFR2), Scalability Setup (NFR4, DR1.2-1.3), Authentication (NFR2.1).
Phase 5: Testing & Deployment	QA Eng, DevOps, All Devs	**40 hrs**	Unit Testing, Integration Testing, Functional Testing (FRs), Performance Testing (NFR1), Security Testing (NFR2), Deployment (DR1.1), CI