

# Jira AI Assistant - Requirements Document

## AI-Powered Intelligent Ticket Management System

**Document Version:** 1.0

**Date:** December 2024

**Status:** Prototype Requirements

**Author:** Development Team

---

## Table of Contents

1. [Executive Summary](#)
  2. [System Architecture](#)
  3. [Core User Flows](#)
  4. [Functional Requirements](#)
  5. [Non-Functional Requirements](#)
  6. [Technology Stack](#)
  7. [Data Models](#)
  8. [User Interface](#)
  9. [Deployment Architecture](#)
  10. [Project Milestones](#)
  11. [Success Metrics](#)
  12. [Security](#)
  13. [API Endpoints](#)
  14. [Future Enhancements](#)
- 

## 1. Executive Summary

### 1.1 Project Overview

**Project Name:** Jira AI Assistant - Intelligent Ticket Management System

**Objective:** Develop an AI-powered automation system that transforms natural language prompts into fully-managed Jira tickets with automatic estimation, intelligent breakdown, and smart assignment based on team capacity and priority.

**Prototype Scope:** MVP (Minimum Viable Product) focusing on core automation flows with learning capabilities.

### 1.2 Key Features

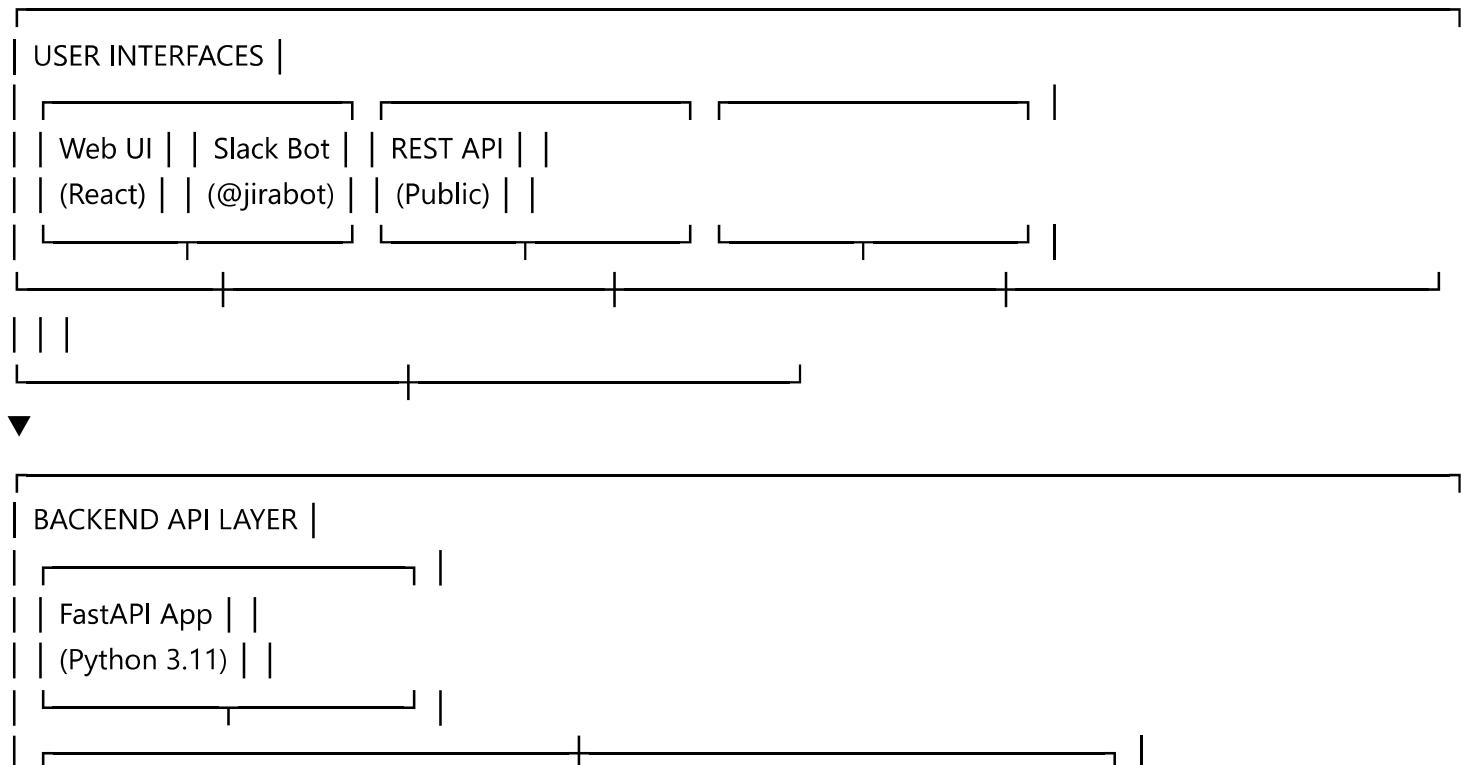
- Natural language story creation
- AI-powered estimation using historical data
- Automatic story breakdown into subtasks
- Intelligent assignment (priority + bandwidth)
- Real-time capacity tracking
- Feedback learning loop
- Multi-channel access (Web, Slack, API)

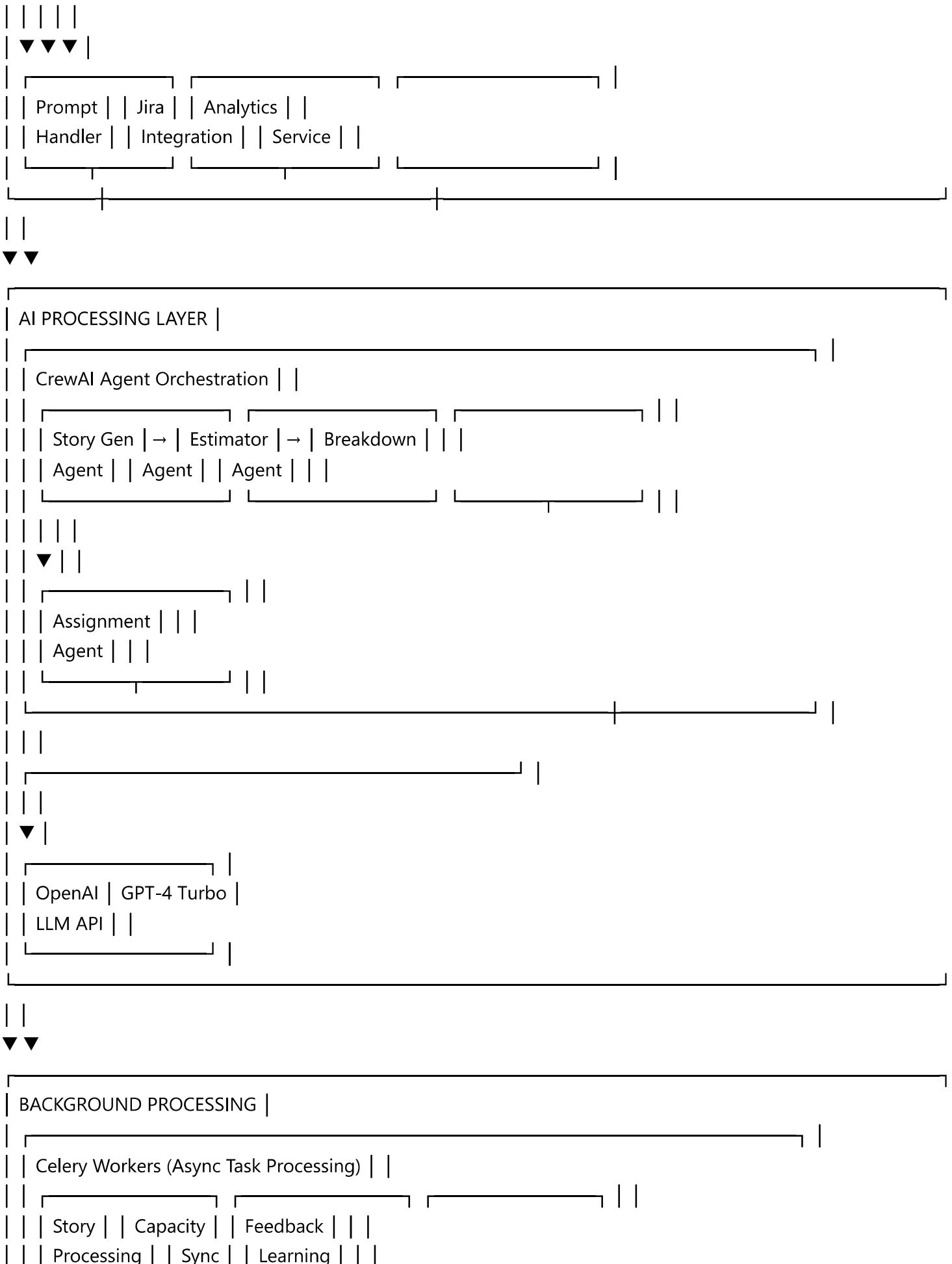
## 1.3 Business Value

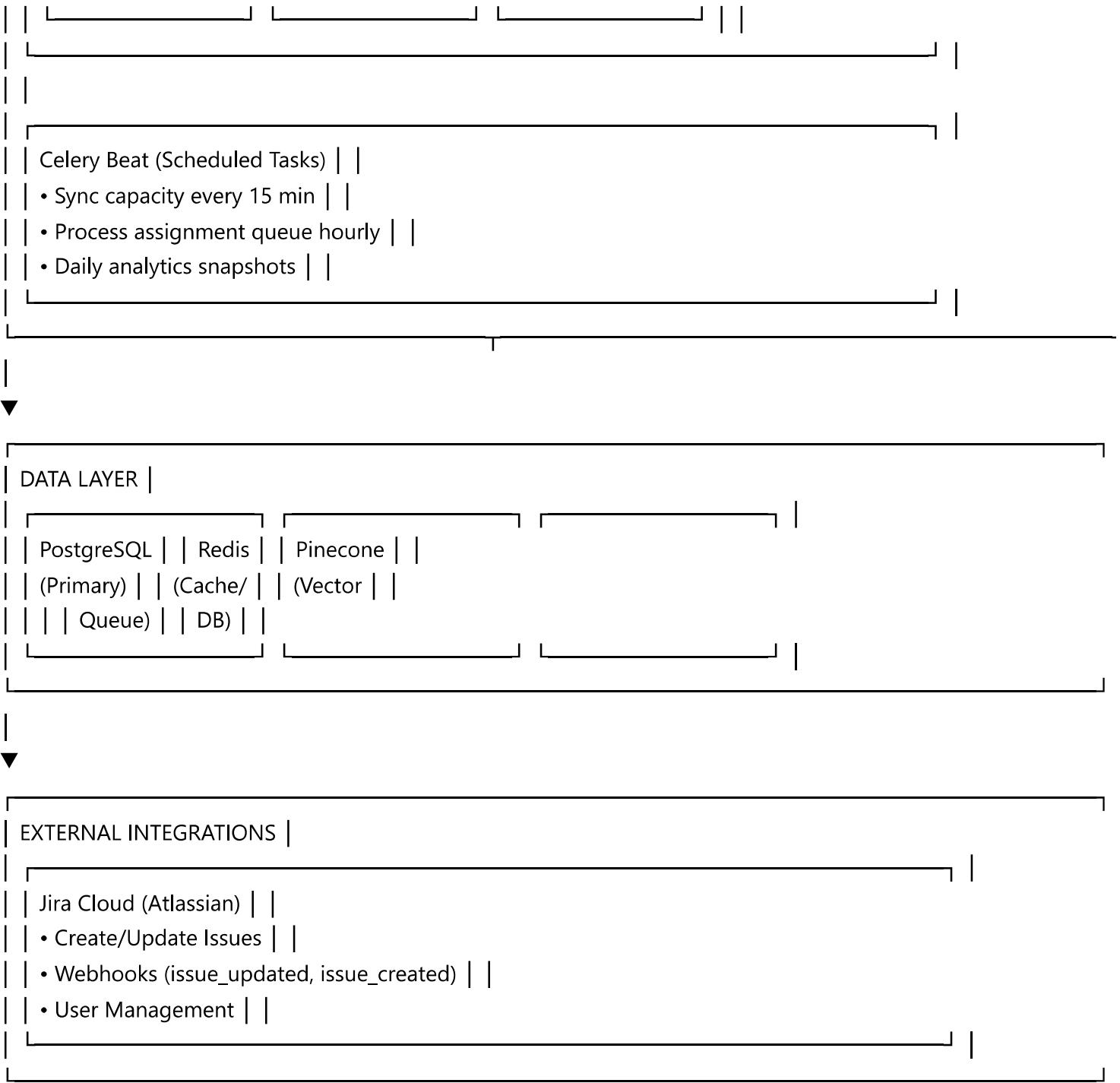
Benefit	Impact
Time Savings	70% reduction in story creation time
Estimation Accuracy	75% accuracy improvement
Workload Balance	Automated capacity-based distribution
Team Productivity	Reduced manual assignment overhead
Knowledge Retention	AI learns from team decisions

## 2. System Architecture

### 2.1 High-Level Architecture







## 2.2 Component Descriptions

### 2.2.1 User Interface Layer

- **Web UI (React):** Primary interface for story creation and monitoring
- **Slack Bot:** Conversational interface for quick story creation
- **REST API:** Programmatic access for integrations

### 2.2.2 Backend API Layer

- **FastAPI**: High-performance async Python web framework
- **Prompt Handler**: Processes natural language inputs
- **Jira Integration**: Bidirectional sync with Jira Cloud
- **Analytics Service**: Performance metrics and insights

### 2.2.3 AI Processing Layer

- **CrewAI Orchestration**: Multi-agent coordination
- **Story Generator Agent**: Creates structured stories from prompts
- **Estimator Agent**: Calculates story points using RAG
- **Breakdown Agent**: Splits large stories into subtasks
- **Assignment Agent**: Matches tickets to team members

### 2.2.4 Background Processing

- **Celery Workers**: Async task execution
- **Celery Beat**: Scheduled periodic tasks
- **Task Types**: Story processing, capacity sync, learning updates

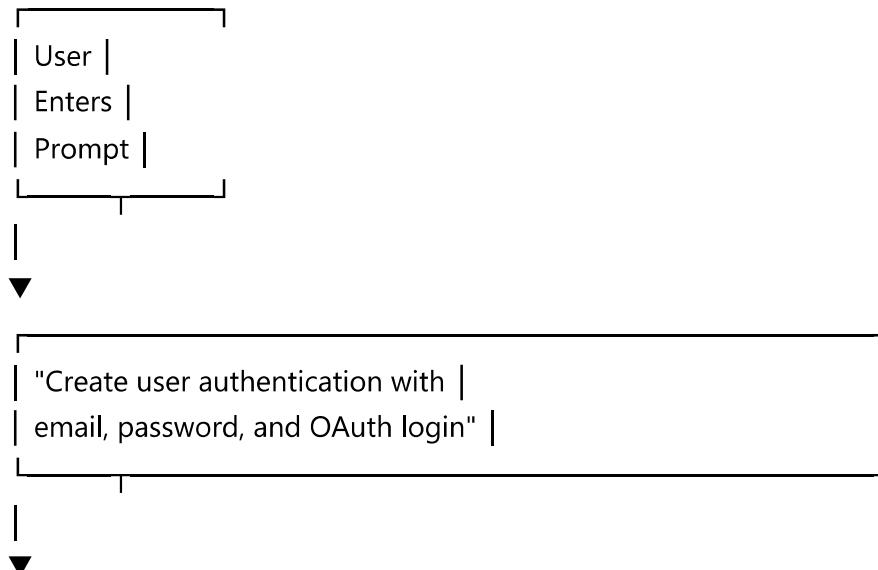
### 2.2.5 Data Layer

- **PostgreSQL**: Relational data (stories, team, feedback)
- **Redis**: Cache, message broker, session storage
- **Pinecone/ChromaDB**: Vector embeddings for RAG

---

## 3. Core User Flows

### 3.1 Story Creation Flow



```
| Story Generation Agent (CrewAI) |
|   • Parse requirements |
|   • Generate title, description |
|   • Create acceptance criteria |
|   • Extract technical requirements |
```

```
| Estimation Agent |
|   • Analyze complexity |
|   • Search similar stories (RAG) |
|   • Estimate story points (Fibonacci) |
|   • Provide reasoning |
```

```
Points > 5? ——————
| |
▼ YES | NO
```

```
| Breakdown Agent |
|   • Frontend tasks |
|   • Backend tasks |
|   • Testing tasks |
|   • Create subtasks |
```

```
| Assignment Agent |
|   • Calculate: Bandwidth × 40% |
|   Skills × 30% |
|   Priority Fit × 20% |
|   Performance × 10% |
|   • Check capacity constraints |
|   • Assign or Queue |
```

```
| Jira API |
```

- | • Create story in Jira |
  - | • Create subtasks (if any) |
  - | • Assign to team member |
  - | • Set story points |
- 



- 
- | User Notification |
  - |  Story PROJ-123 created |
  - | Estimated: 8 points |
  - | Assigned to: John Doe |
  - | View in Jira |
- 

### 3.2 Feedback Learning Loop

- 
- | Human |
  - | Changes |
  - | in Jira |
- 



- 
- | Jira Webhook |
  - | Event: issue\_updated |
  - | • Estimation changed: 5 → 8 |
  - | • Assignee changed: John → Jane |
- 



- 
- | Feedback Service |
  - | • Detect what changed |
  - | • Calculate error/difference |
  - | • Store in feedback tables |
- 



- 
- | Learning Analysis |

- | • Identify patterns |
  - | • Update prompts dynamically |
  - | • Adjust scoring weights |
  - | • Update vector DB with corrections |
- 



- 
- | Next Estimation (Improved!) |
  - | • Uses learned adjustments |
  - | • References corrected data |
  - | • Higher accuracy |
- 

### 3.3 Assignment Decision Flow

- 
- | New Ticket |
  - | PROJ-456 |
  - | Priority: High |
  - | Points: 8 |
- 

- 
- | Get Eligible Candidates |
  - | • Not OOO |
  - | • Has capacity |
  - | • Has required skills |
- 

- 
- | Score Each Candidate |
  - | |
  - | John: 87/100 |
  - | • Bandwidth: 70 |
  - | • Skills: 95 |
  - | • Priority Fit: 100 |
  - | • Performance: 85 |
  - | |
  - | Jane: 92/100 ← Winner |

- Bandwidth: 100 |
  - Skills: 100 |
  - Priority Fit: 85 |
  - Performance: 80 |
- 
- ▼

- Validate Assignment |
  - Won't exceed capacity?  |
  - Under max tickets?  |
- 
- ▼

- Assign in Jira |
  - Update assignee |
  - Update team capacity |
  - Log assignment reasoning |
- 

## 4. Functional Requirements

### 4.1 Natural Language Story Creation

ID	Requirement	Priority	Acceptance Criteria
FR-1.1	Accept natural language prompt (min 10 chars)	High	System accepts prompts ≥10 characters
FR-1.2	Generate story title in user story format	High	Title follows "As a [user], I want [feature]"
FR-1.3	Create detailed description with context	High	Description includes problem, solution, impact
FR-1.4	Generate 3-7 acceptance criteria	High	List of testable criteria provided
FR-1.5	Extract technical requirements automatically	Medium	Technical notes section populated

ID	Requirement	Priority	Acceptance Criteria
FR-1.6	Identify required skills from description	Medium	Skills array returned (e.g., ["Python", "React"])

## 4.2 AI-Powered Estimation

ID	Requirement	Priority	Acceptance Criteria
FR-2.1	Estimate using Fibonacci scale (1,2,3,5,8,13,21)	High	Only valid Fibonacci numbers returned
FR-2.2	Use RAG to find similar historical stories	High	Queries vector DB with 5 similar stories
FR-2.3	Provide estimation reasoning	Medium	Text explanation of estimation logic
FR-2.4	Include confidence score	Low	Percentage confidence (e.g., 75%)

## 4.3 Intelligent Story Breakdown

ID	Requirement	Priority	Acceptance Criteria
FR-3.1	Auto-break stories > 5 points into subtasks	High	Triggered when estimated_points > 5
FR-3.2	Create 4-8 subtasks per story	High	Number of subtasks in range
FR-3.3	Categorize tasks (Frontend, Backend, Testing, etc.)	Medium	Each subtask has category label
FR-3.4	Link subtasks to parent story in Jira	High	Parent-child relationship created

## 4.4 Smart Assignment (Priority + Bandwidth)

ID	Requirement	Priority	Acceptance Criteria
FR-4.1	Calculate assignment score based on 4 factors	High	Score = Bandwidth×40% + Skills×30% + Priority×20% + Performance×10%
FR-4.2	Never exceed team member max capacity	High	Assignment rejected if over capacity
FR-4.3	Assign high priority to senior developers	High	Priority="High" filters out Junior devs
FR-4.4	Queue tickets when no capacity available	Medium	Ticket added to assignment queue

ID	Requirement	Priority	Acceptance Criteria
FR-4.5	Respect out-of-office status	High	OOO members excluded from candidates
FR-4.6	Balance workload across team	Medium	No member gets >30% more work than average

## 4.5 Capacity Management

ID	Requirement	Priority	Acceptance Criteria
FR-5.1	Sync capacity from Jira every 15 minutes	High	Celery beat task runs on schedule
FR-5.2	Track current story points per developer	High	Database field updated on assignment
FR-5.3	Track concurrent ticket count	High	Count of open tickets per member
FR-5.4	Calculate availability percentage	Medium	(Available / Max Capacity) × 100
FR-5.5	Mark members as OOO	Medium	API endpoint to set OOO dates

## 4.6 Feedback & Learning

ID	Requirement	Priority	Acceptance Criteria
FR-6.1	Capture manual estimation changes via webhook	High	Webhook handler processes issue_updated
FR-6.2	Capture manual assignment changes	High	Reassignment logged in FeedbackAssignment table
FR-6.3	Store actual completion time	Medium	Calculated from created → resolved dates
FR-6.4	Update vector DB with completed stories	High	Embeddings added on story completion
FR-6.5	Adjust AI prompts based on patterns	Medium	Dynamic prompt adjustments applied
FR-6.6	Show estimation accuracy metrics	Low	Dashboard displays accuracy percentage

## 4.7 Multi-Channel Access

ID	Requirement	Priority	Acceptance Criteria
FR-7.1	Web UI for story creation	High	React app with form interface
FR-7.2	Slack bot integration	Medium	Bot responds to mentions and commands
FR-7.3	REST API for programmatic access	High	OpenAPI spec with all endpoints
FR-7.4	Real-time status updates	Low	WebSocket or polling for status

## 5. Non-Functional Requirements

### 5.1 Performance

ID	Requirement	Target	Measurement Method
NFR-1.1	Story creation end-to-end time	< 10 seconds	API response time logging
NFR-1.2	API response time (95th percentile)	< 2 seconds	APM monitoring
NFR-1.3	Capacity sync completion time	< 30 seconds for 50 users	Celery task duration
NFR-1.4	Concurrent users supported	100 users	Load testing
NFR-1.5	Database query response time	< 500ms	Query logging

### 5.2 Reliability

ID	Requirement	Target	Measurement Method
NFR-2.1	System uptime	99.5%	Uptime monitoring
NFR-2.2	Webhook delivery retry	3 attempts with exponential backoff	Retry logic implementation
NFR-2.3	Queue durability	Persist failed assignments	Redis persistence enabled
NFR-2.4	Data backup	Daily automated backups	Backup script scheduled
NFR-2.5	Error recovery	Graceful degradation on AI failure	Fallback to default values

### 5.3 Security

ID	Requirement	Implementation
NFR-3.1	API authentication	JWT tokens or API keys
NFR-3.2	Jira credentials storage	Encrypted environment variables
NFR-3.3	Transport security	HTTPS only (production)
NFR-3.4	Rate limiting	100 requests/minute per user
NFR-3.5	Input validation	Pydantic models for all inputs
NFR-3.6	SQL injection prevention	ORM (SQLAlchemy) parameterized queries

## 5.4 Scalability

ID	Requirement	Target
NFR-4.1	Team size supported	Up to 50 members (prototype)
NFR-4.2	Story creations per day	500 stories/day
NFR-4.3	Historical data retention	6 months
NFR-4.4	Database growth	Plan for 10GB in 6 months
NFR-4.5	Horizontal scaling	Support multiple worker instances

## 5.5 Usability

ID	Requirement	Acceptance Criteria
NFR-5.1	Prompt input simplicity	No special syntax required
NFR-5.2	Assignment reasoning visibility	Clear explanation shown to users
NFR-5.3	Error message quality	Actionable error messages
NFR-5.4	Dashboard load time	< 3 seconds
NFR-5.5	Mobile responsiveness	UI works on tablets (optional for prototype)

## 6. Technology Stack

### 6.1 Backend Technologies

Programming Language: Python 3.11

Web Framework: FastAPI 0.104+

- Async/await support
- Automatic OpenAPI documentation
- Pydantic validation
- High performance (ASGI)

Agent Framework: CrewAI 0.1+

- Built on LangChain
- Multi-agent orchestration
- Role-based agent design
- Task delegation support

Async Task Processing: Celery 5.3+

- Redis broker
- Flower monitoring UI
- Beat scheduler for periodic tasks
- Result backend for task tracking

ORM & Database: SQLAlchemy 2.0+

- Async support
- Alembic for migrations
- PostgreSQL adapter (psycopg2)

LLM Provider: OpenAI

- Model: GPT-4 Turbo (gpt-4-turbo-preview)
- API: openai>=1.3.0
- Embeddings: text-embedding-3-small

Alternative LLM: Anthropic Claude 3

- Model: claude-3-opus-20240229
- API: anthropic>=0.7.0
- Use case: Fallback or comparison

Vector Database Options:

Primary (Cloud): Pinecone

- Managed service
- Auto-scaling
- Client: pinecone-client>=2.2.4

Alternative (Self-hosted): ChromaDB

- Docker container
- Local development
- Client: chromadb>=0.4.18

RAG Framework: LangChain

- Vector store integration
- Document loaders
- Embeddings wrapper
- Version: langchain>=0.0.335

**Primary Database: PostgreSQL 15**

- Relational data storage
- JSON/JSONB support
- Full-text search
- Connection pooling

**Cache & Message Broker: Redis 7**

- Celery broker
- Session storage
- Rate limiting
- Pub/Sub for real-time updates

**Vector Storage: Pinecone / ChromaDB**

- Story embeddings
- Similarity search
- RAG context retrieval

**Framework: React 18**

- Functional components
- Hooks API
- Context for state management

**Build Tool: Vite**

- Fast HMR (Hot Module Replacement)
- Optimized production builds
- ESM support

**Styling: Tailwind CSS 3**

- Utility-first CSS
- Responsive design
- Custom components

**HTTP Client: Axios**

- Promise-based requests
- Interceptors for auth
- Request/response transformation

**UI Components (Optional):**

- Headless UI (Tailwind official)
- React Icons
- Recharts (for analytics)

**Jira Integration:**

Library: `jira-python (jira>=3.5.2)`

Alternative: `atlassian-python-api>=3.41.0`

Authentication: API Token (Basic Auth)

**Features:**

- Issue CRUD operations
- Custom fields access
- Webhook registration
- User management

- JQL search

#### Slack Integration:

Library: slack-bolt>=1.18.0

Mode: Socket Mode (no public URL needed)

#### Features:

- App mentions
- Slash commands
- Interactive components
- Message formatting

#### Monitoring:

Celery: Flower (web-based monitoring)

Logging: Python logging module

Metrics (Future): Prometheus + Grafana

#### Containerization:

- Docker 24+
- Docker Compose 2.20+
- Multi-stage builds
- Health checks

#### Development:

- python-dotenv for environment management
- pytest for testing
- black for code formatting
- flake8 for linting

#### Version Control:

- Git
- GitHub / GitLab

#### CI/CD (Future):

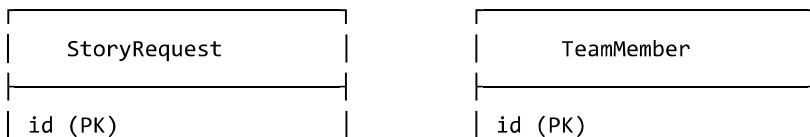
- GitHub Actions
- Docker Hub / Container Registry
- Automated testing
- Deployment pipelines

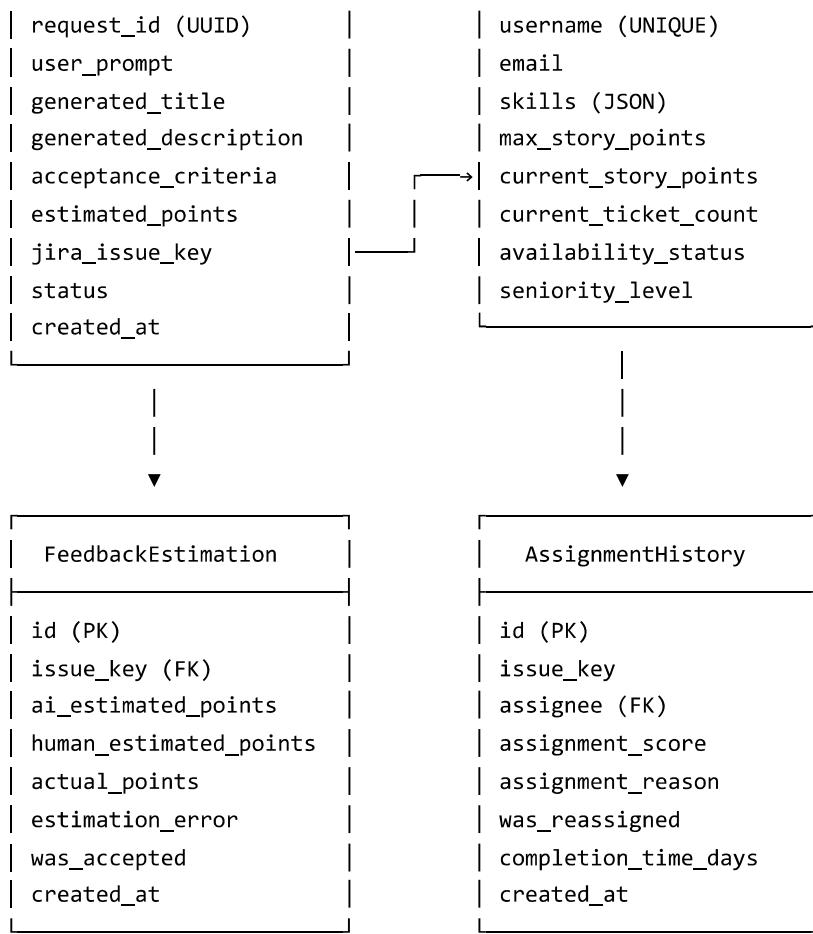
#### Production (Future):

- AWS ECS / EKS
- GCP Cloud Run
- Azure Container Instances
- Managed databases (RDS, Cloud SQL)

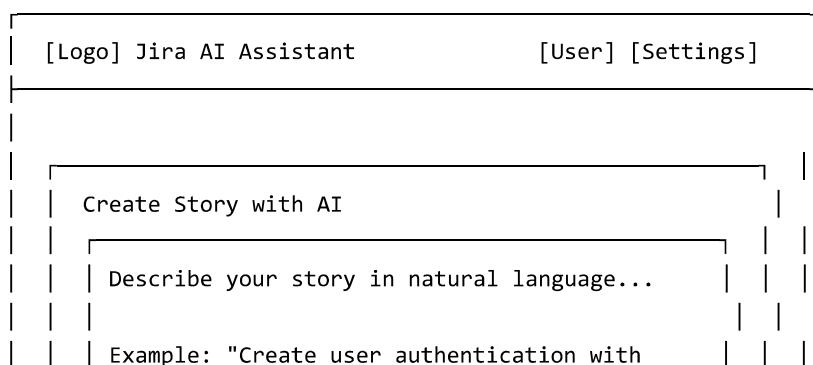
## 7. Data Models

### 7.1 Entity Relationship Diagram





## User Interface



email, password, and Google OAuth"

Issue Type: [Story ▼] Priority: [Medium ▼]

[ ] Auto-estimate story points  
[ ] Auto-break into subtasks (if > 5 points)  
[ ] Auto-assign to team members

[Create Story →]

Quick Stats

Team Capacity	Available Capacity	Utilization
250 pts	70 pts	

Stories Created	Estimation Accuracy	Assignment Accuracy
45	78%	82%

Recent Stories [View All →]

<input checked="" type="checkbox"/> PROJ-123 User Authentication [8 pts] Assigned to: @john.doe Created: 2 hours ago [View in Jira] [Show Details]
<input type="checkbox"/> PROJ-124 Export CSV Feature [5 pts] Assigned to: @jane.smith Created: 5 hours ago [View in Jira] [Show Details]

## Story Creation Result Screen

Story Created Successfully!

PROJ-123

Title:

| As a user, I want to authenticate with email and  
| password so that I can securely access my account | |

| Estimated: 8 story points | |

| Priority: High | |

| Assigned to: John Doe (@john.doe) | |

---

| Acceptance Criteria: | |

| 1. User can register with valid email | |

| 2. Password must meet security requirements | |

| 3. User receives confirmation email | |

| 4. User can login with credentials | |

| 5. Failed login shows appropriate error | |

| Subtasks Created (4): | |

| • PROJ-123-1: Design login UI (Frontend) | |

| • PROJ-123-2: Implement auth API (Backend) | |

| • PROJ-123-3: Write authentication tests (Testing) | |

| • PROJ-123-4: Update API documentation (Docs) | |

| Assignment Reasoning: | |

| Has good available capacity (10/20 pts used) | |

| Strong skills match (Python, Authentication) | |

| Appropriate seniority for High priority | |

| Strong historical performance (8.5/10) | |

| [View in Jira →] [Create Another] [Done] | |

### 8.3 Team Capacity Dashboard

| Team Capacity Overview [Refresh ⏪] | |

| Status Distribution | |

| Available (3) Busy (2) Overloaded (1) | |

| Team Members | |

| John Doe (Senior Backend Developer) | |

| 15/20 pts (75%) | |

| Active Tickets: 5 | |

| High: 2 Medium: 2 Low: 1 | |

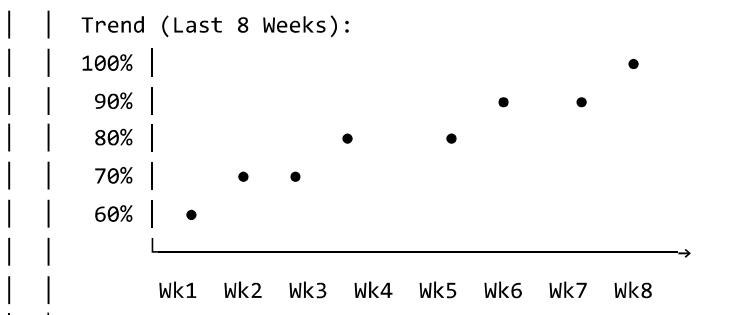
| Skills: Python, FastAPI, PostgreSQL, AWS | |

| Avg Completion: 3.5 days Quality: 8.5/10 | |

Status: <span style="color: orange;">⚠️ Busy</span>	
<span style="color: purple;">👤</span> Jane Smith (Mid Frontend Developer)	
18/20 pts (90%)	
<span style="color: brown;">📦</span> Active Tickets: 4	
<span style="color: pink;">🎯</span> High: 1 Medium: 2 Low: 1	
<span style="color: darkblue;">💼</span> Skills: React, TypeScript, Tailwind CSS	
<span style="color: green;">📊</span> Avg Completion: 4.2 days Quality: 7.8/10	
Status: <span style="color: orange;">⚠️ Busy</span>	
<span style="color: purple;">👤</span> Bob Wilson (Senior DevOps Engineer)	
0/20 pts (0%)	
<span style="color: brown;">📦</span> Active Tickets: 0	
<span style="color: darkblue;">💼</span> Skills: AWS, Docker, Kubernetes, CI/CD	
<span style="color: blue;">🏖️</span> Out of Office until Dec 20, 2024	
Status: <span style="color: red;">🔴 OOO (Vacation)</span>	
<span style="color: purple;">👤</span> Alice Johnson (Junior Full-Stack)	
5/15 pts (33%)	
<span style="color: brown;">📦</span> Active Tickets: 2	
<span style="color: pink;">🎯</span> Medium: 1 Low: 1	
<span style="color: darkblue;">💼</span> Skills: Python, React, PostgreSQL	
<span style="color: green;">📊</span> Avg Completion: 5.8 days Quality: 7.2/10	
Status: <span style="color: green;">✓ Available</span>	
Assignment Queue (2 tickets waiting) [Process Now]	
PROJ-125   High Priority   8 pts	
Reason: All candidates would be overloaded	
Waiting: 2 hours	
PROJ-126   Medium Priority   5 pts	
Reason: No team members with required skills	
Waiting: 5 hours	

#### 8.4 Analytics Dashboard

Analytics & Insights	[Last 30 Days ▼]
Estimation Accuracy	
Overall Acceptance Rate: 78%	
(AI estimates not changed by humans)	
Average Error: ±1.2 points	



## Assignment Accuracy

Acceptance Rate: 82%  
(Assignments not manually changed)

Reassignment Patterns (Top 3):

1. john.doe → jane.smith (5 times)  
Reason: Frontend tasks reassigned
2. alice.johnson → john.doe (3 times)  
Reason: Complexity too high for Junior
3. jane.smith → alice.johnson (2 times)  
Reason: Growth opportunity

## Learning Insights

- 💡 AI is learning:
  - Authentication stories now estimated higher  
(was underestimating by 30%)
  - Frontend tasks preferred for @jane.smith
  - Testing tasks taking 20% longer than estimated
- 🎯 Recommendations:
  - Consider hiring more backend capacity
  - @alice.johnson ready for Medium priority tasks
  - Update sprint capacity to 18 pts/person

## 9. Deployment Architecture

### 9.1 Development Environment (Docker Compose)

#### Docker Compose - Local Development

Container: api                          Port: 8000  
- FastAPI application  
- Hot reload enabled  
- Mounted volume: ./backend:/app

```

| Container: worker          (internal) |
| - Celery worker (4 concurrency) |
| - Handles async tasks      |
|
| Container: beat           (internal) |
| - Celery beat scheduler   |
| - Periodic tasks (capacity sync, queue processing) |
|
| Container: flower          Port: 5555 |
| - Celery monitoring UI    |
| - Real-time task tracking |
|
| Container: frontend        Port: 3000 |
| - React development server |
| - Hot module replacement   |
| - Mounted volume: ./frontend/src:/app/src |
|
| Container: postgres         Port: 5432 |
| - PostgreSQL 15            |
| - Volume: postgres_data:/var/lib/postgresql/data |
| - Health check enabled     |
|
| Container: redis            Port: 6379 |
| - Redis 7 (cache + broker) |
| - Volume: redis_data:/data |
| - Persistence enabled       |
|
| Container: chromadb         Port: 8001 |
| - Vector database           |
| - Volume: chroma_data:/chroma |
|
| Container: slack-bot        (internal) |
| - Slack integration (optional) |
| - Socket mode (no public URL) |
|
| Network: jira-ai-network (bridge) |
| Volumes: postgres_data, redis_data, chroma_data |

```

### 13. API Endpoints

**POST /api/prompt/create-story**  
 Description: Create Jira story from natural language prompt  
 Request Body:

```
{
  "prompt": "string (required, min 10 chars)",
  "issue_type": "Story|Task|Bug (default: Story)",
  "priority": "Highest|High|Medium|Low (default: Medium)",
  "project_key": "string (optional)",
  "epic_key": "string (optional)",
  "labels": ["string"] (optional),
```

```
"auto_breakdown": boolean (default: true),
"auto_estimate": boolean (default: true),
"auto_assign": boolean (default: true)
}
```

Response: 200 OK

```
{
  "request_id": "uuid",
  "status": "processing|completed|failed",
  "generated_story": {
    "title": "string",
    "description": "string",
    "acceptance_criteria": ["string"],
    "estimated_points": integer
  },
  "jira_issue_key": "string (if completed)",
  "jira_url": "string (if completed)",
  "created_at": "datetime"
}
```

---

GET /api/prompt/story-status/{request\_id}

Description: Get status of story creation request

Parameters:

- request\_id: UUID (path parameter)

Response: 200 OK

```
{
  "request_id": "uuid",
  "status": "pending|processing|completed|failed",
  "generated_story": {...},
  "jira_issue_key": "string",
  "error_message": "string (if failed)"
}
```

---

POST /api/prompt/chat

Description: Conversational interface for story creation

Request Body:

```
{
  "message": "string",
  "session_id": "string (optional)",
  "context": {} (optional)
}
```

Response: 200 OK

```
{
  "response": "string",
  "suggestions": ["string"],
  "actions": [
    {
      "type": "create_story",

```

```
"data": {...}
},
],
"session_id": "string"
}

---  
  
POST /api/prompt/suggest-estimation
Description: Get estimation suggestion for existing story
Request Body:
{
  "story_title": "string",
  "story_description": "string"
}
Response: 200 OK
{
  "estimated_points": integer,
  "reasoning": "string",
  "confidence": float,
  "similar_stories": [...]
}  
  
13.2 Capacity Management
GET /api/capacity/team
Description: Get team capacity overview
Response: 200 OK
{
  "total_team_capacity": integer,
  "total_used_capacity": integer,
  "available_capacity": integer,
  "utilization_percentage": float,
  "team_size": integer,
  "available_members": integer,
  "members_by_status": {
    "available": ["username"],
    "busy": ["username"],
    "overloaded": ["username"]
  }
}  
  
---  
  
GET /api/capacity/member/{username}
Description: Get individual member capacity
Parameters:
- username: string (path parameter)
Response: 200 OK
{
  "username": "string",
  "display_name": "string",
```

```

    "seniority": "string",
    "current_story_points": integer,
    "max_story_points": integer,
    "available_story_points": integer,
    "current_ticket_count": integer,
    "availability_percentage": float,
    "availability_status": "available|busy|overloaded|ooo",
    "is_out_of_office": boolean,
    "skills": ["string"],
    "preferred_work": ["string"]
}

```

---

**POST /api/capacity/mark-ooo**

Description: Mark team member as out of office

Request Body:

```
{
  "username": "string",
  "start_date": "datetime",
  "end_date": "datetime",
  "reason": "string",
  "partial_capacity": float (optional, 0-100)
}
```

Response: 200 OK

```
{
  "status": "success",
  "message": "string"
}
```

### 13.3 Assignment

**POST /api/assignment/assign-ticket**

Description: Manually trigger ticket assignment

Request Body:

```
{
  "issue_key": "string",
  "priority": "string",
  "estimated_points": integer,
  "required_skills": ["string"]
}
```

Response: 200 OK

```
{
  "assigned_to": "string",
  "display_name": "string",
  "assignment_score": float,
  "reasoning": "string",
  "alternatives": [
    {
      "username": "string",
      "score": float
    }
  ]
}
```

```
    }
]
}

---  
  
GET /api/assignment/queue
Description: Get current assignment queue
Response: 200 OK
{
  "queued_count": integer,
  "items": [
    {
      "issue_key": "string",
      "priority": "string",
      "estimated_points": integer,
      "attempts": integer,
      "reason": "string"
    }
  ]
}
```

### 13.4 Analytics

```
GET /api/analytics/estimation-accuracy
Description: Get estimation accuracy metrics
Response: 200 OK
{
  "acceptance_rate": float,
  "average_error": float,
  "total_estimations": integer,
  "monthly_trend": [
    {
      "month": "string",
      "error": float,
      "count": integer
    }
  ]
}
```

```
---  
  
GET /api/analytics/assignment-accuracy
Description: Get assignment accuracy metrics
Response: 200 OK
{
  "acceptance_rate": float,
  "total_assignments": integer,
  "reassignments": integer,
  "common_reassignment_patterns": [
    {

```

```
        "from": "string",
        "to": "string",
        "count": integer
    }
]
}
```

### 13.5 Webhooks

POST /api/webhook/jira

Description: Jira webhook handler (internal)

Request Body: Jira webhook payload

Response: 200 OK

```
{
  "status": "received|processed|ignored"
}
```