OwlTask Software Requirements Document

Project Description

What's the Problem?

In today's collaborative work environment, teams struggle with fragmented task management that leads to missed deadlines, unclear responsibilities, and poor coordination. Current solutions either lack real-time collaboration features or are overly complex for small to medium teams. The fundamental problem is that team members lose visibility into project progress and struggle to understand how their individual contributions fit into the larger picture.

Target Audience: Small to medium-sized teams (5-20 members) including: - Project managers who need oversight of team progress - Team members who need clarity on their responsibilities and deadlines - Cross-functional teams requiring real-time coordination

Why This Matters: Poor task coordination leads to project delays, duplicated effort, and team frustration. A streamlined, real-time task management system can significantly improve team productivity and project success rates.

How Will OwlTask Address the Problem?

OwlTask provides a unified platform for real-time task coordination with intelligent status summaries. Unlike traditional task managers that simply list tasks, OwlTask focuses on **contextual awareness** - helping teams understand not just what needs to be done, but how everything connects and progresses together.

Unique Approach: Integration of AI-powered status summaries that provide natural language insights into project progress, making it easy for team members to quickly understand "where we are" without diving into detailed task lists.

Key Advantages: - Real-time synchronization eliminates information lag - Natural language status queries reduce cognitive load - Lightweight design ensures quick adoption - Extensible architecture supports future growth

Potential Disadvantages: - Requires consistent internet connectivity for real-time features - AI summaries depend on external API availability

Use Case Descriptions

Primary Use Cases

UC1: Task Creation and Assignment - **Actor**: Project Manager/Team Lead - **Goal**: Create and assign tasks to team members - **Scenario**: Manager creates a task with description, priority, due date, and assigns to specific team member or group - **Success Criteria**: Task is created, assigned user receives notification, task appears in relevant views

UC2: Real-time Status Updates - **Actor**: Team Member - **Goal**: Update task progress and see current team status - **Scenario**: User updates their task status, system broadcasts changes to all connected team members - **Success Criteria**: All team members see updated status immediately, no data conflicts

UC3: Intelligent Progress Inquiry - **Actor**: Project Manager/Team Member - **Goal**: Get contextual summary of project progress - **Scenario**: User asks "How's progress since Monday?" and receives AI-generated summary of completed tasks, upcoming deadlines, and current blockers - **Success Criteria**: User receives relevant, accurate summary within 5 seconds

UC4: Group Coordination - **Actor**: Team Member - **Goal**: Coordinate work within assigned groups - **Scenario**: User views group tasks, sees what teammates are working on, identifies dependencies - **Success Criteria**: Clear visibility into group progress and individual contributions

UC5: Deadline Management - **Actor**: All Users - **Goal**: Stay aware of approaching deadlines - **Scenario**: System proactively notifies users of upcoming deadlines and overdue tasks - **Success Criteria**: No missed deadlines due to lack of awareness

Functional Requirements

Core Functional Requirements (Must Have - 5 Week Scope)

FR1: Basic User Management - Simple user registration and login (no federated authentication) - Basic user profiles (name, email only) - Create and join groups (flat structure only)

FR2: Essential Task Management - Create, read, update, delete tasks - Basic task properties: description, due date, priority (high/medium/low), status - Task assignment to individuals or groups - Task status tracking (not started, in progress, completed)

FR3: Real-time Synchronization - WebSocket-based real-time updates for task changes - Automatic synchronization across connected clients - Basic conflict resolution (last-write-wins)

FR4: AI-Powered Status Summaries (Unique Feature) - Natural language query processing ("How's progress since Monday?") - Integration with GPT-4o for intelligent summaries - Context-aware progress reporting based on recent task activity

FR5: Basic Notification System - Browser-based notifications for task assignments - Simple deadline reminders - Real-time alerts for task status changes

Secondary Functional Requirements (Should Have)

FR6: Basic Reporting - Task completion statistics - Individual and group progress views - Simple dashboard with key metrics

FR7: Search and Filtering - Search tasks by keyword - Filter by assignee, status, due date, priority - Group-based task views

Non-Functional Requirements

NFR1: Performance - Page load time $<$ 3 seconds - Real-time updates delivered within 1 second - Support for up to 20 concurrent users (small team focus)

NFR2: Usability - Responsive design for desktop and mobile - Intuitive interface requiring minimal training - Clean, modern UI focused on core functionality

NFR3: Reliability - Basic error handling and graceful degradation - Data persistence - 95% uptime target (realistic for 5-week project)

NFR4: Security - Basic user authentication - Input validation and sanitization - HTTPS in production

Description of Omitted Items

Major Feature Categories Omitted from PROJECT_DESCRIPTION

User Management Omissions: - **System maintainer/developer roles**: Only general users supported - *Rationale*: Admin features not critical for core task coordination - *Implementation Effort*: Medium (2 weeks) - **Complex group hierarchies**: Only flat group structure - *Rationale*: Single-line vs multi-line hierarchies add significant complexity - *Implementation Effort*: High (3-4 weeks) - **Advanced authorization system**: No whitelisting/blacklisting, hierarchical permissions, or inherited permissions - *Rationale*: Basic group membership sufficient for small teams - *Implementation Effort*: High (4-5 weeks) - **Federated authentication**: No Google/ADRICE login integration - *Rationale*: Simple login adequate for prototype - *Implementation Effort*: Medium (1-2 weeks)

Task Management Omissions: - **Task categorization**: No area/sprint/milestone categories or hierarchical categories - *Rationale*: Simple task lists meet 80% of coordination needs - *Implementation Effort*: Medium (2-3 weeks) - **Composite tasks**: No sub-tasks or task hierarchies - *Rationale*: Adds UI and data model complexity without core value - *Implementation Effort*: High (3-4 weeks) - **Assignment restrictions**: No location, role, or permission-based assignment limits - *Rationale*: Small teams don't need complex assignment rules - *Implementation Effort*: Medium (2 weeks) - **Resource projection**: No projected resources needed tracking - *Rationale*: Focus on completion rather than resource planning - *Implementation Effort*: Medium (2 weeks) - **Advanced**

filtering: No complex filtering by categories, groups, etc. - *Rationale*: Basic search sufficient for small task volumes - *Implementation Effort*: Low (1 week)

Communication Omissions: - **Task-specific chat**: No discussion threads on individual tasks - *Rationale*: Teams can use external chat tools initially - *Implementation Effort*: High (3-4 weeks) - **Group discussions**: No dedicated discussion areas - *Rationale*: Not core to task coordination - *Implementation Effort*: Medium (2-3 weeks) - **Archived and searchable communications**: No communication history - *Rationale*: External tools available for team communication - *Implementation Effort*: High (4+ weeks)

Advanced Features Omitted: - **Complex notification system**: No offline notification queuing or advanced notification preferences - *Rationale*: Browser notifications sufficient for real-time coordination - *Implementation Effort*: Medium (2 weeks) - **Advanced scalability**: No multi-server deployment or advanced caching - *Rationale*: Single server adequate for small teams - *Implementation Effort*: High (4+ weeks) - **Mobile native apps**: Web responsive design only - *Rationale*: Mobile web sufficient for 5-week timeline - *Implementation Effort*: High (6+ weeks)

Strategic Omissions Rationale

Focus on Core Mission: OwlTask's fundamental goal is real-time task coordination with intelligent status insights. Every omitted feature, while potentially valuable, doesn't directly contribute to this core mission within our 5-week constraint.

80/20 Rule Applied: The included features address 80% of small team coordination needs with 20% of the possible implementation effort. Complex permission systems, hierarchical tasks, and advanced communication features serve edge cases that don't justify their development cost.

Extensibility Preserved: The modular MVVM architecture with clear separation of concerns makes it straightforward to add omitted features in future releases: - Observer pattern ready for complex notification systems - Strategy pattern enables pluggable authorization schemes - Composite pattern prepared for hierarchical task structures - Repository pattern supports advanced filtering and search

Implementation Effort Estimates

Total Omitted Effort: ~35-50 weeks of development **Core Implementation**: ~4-5 weeks **Effort Ratio**: We're implementing roughly 10% of possible features to deliver 80% of core value

This focused approach ensures we can deliver a polished, working application that solves the fundamental coordination problem rather than a feature-rich but incomplete system.

Infrastructure Requirements

Technology Stack Selection

Frontend: React with TypeScript - **Rationale**: Provides robust component-based architecture with type safety - **Advantages**: Large ecosystem, excellent real-time capabilities, strong community support - **Disadvantages**: Learning curve for team members unfamiliar with React

Backend: Java with Spark Framework - **Rationale**: Lightweight, suitable for microservices, excellent WebSocket support - **Advantages**: Fast development, easy deployment, good performance for small-medium scale - **Disadvantages**: Less feature-rich than Spring Boot, smaller community

Database: PostgreSQL - **Rationale**: Reliable relational database with good performance characteristics - **Advantages**: ACID compliance, excellent JSON support, mature ecosystem - **Disadvantages**: Requires more setup than NoSQL alternatives

Deployment: Heroku with Docker - **Rationale**: Simplified deployment process, good for prototyping and small-scale production - **Advantages**: Easy scaling, integrated CI/CD, minimal DevOps overhead - **Disadvantages**: Higher cost at scale, vendor lock-in

External Services: OpenAI GPT-4o API - **Rationale**: Provides advanced natural language processing capabilities - **Advantages**: State-of-the-art AI capabilities, well-documented API - **Disadvantages**: External dependency, API costs, rate limiting

Architecture Decisions

MVVM Pattern: Separates presentation logic from business logic, enabling better testability and maintainability.

WebSocket Communication: Enables real-time updates without polling overhead.

RESTful API Design: Provides clear, standard interface for all CRUD operations.

Containerized Deployment: Ensures consistent environments and simplified scaling.

Summary

OwlTask addresses the critical need for real-time task coordination in small to medium teams by combining traditional task management with AI-powered insights. The requirements focus on delivering core value quickly while maintaining extensibility for future enhancements. The selected technology stack

balances development speed with scalability, ensuring the application can grow with user needs while remaining maintainable and cost-effective.