Jira Software Clone: SDLC Document

By Bethusile Mafumana

1. Planning

Project Charter and Timelines			
Project Name:	Jira Software Clone – Project Management Tool		
Objective:	To design and build a full-stack web application that replicates the core functionality of Jira, enabling teams to manage projects, assign tasks, track progress, and collaborate visually.		
Start Date:	May 2025		
Projected Completion:	August 2025		
Development Methodology:	Agile (Scrum-based iterations)		
Feasibility Study			
1. Technical feasibility	High. The tech stack (ASP.NET MVC, Entity Framework, SQL Server, HTML/CSS/JS) is well-suited for enterprise web applications.		
2. Operational feasibility	High. The application directly supports modern project management needs.		
3. Economic feasibility	Cost-effective, as development uses open-source tools and self-hosted environments initially. Potential for cloud deployment later.		
4. Legal feasibility	No IP conflicts with Jira as this is a learning/personal project without commercial intent.		
Risk Analysis			
Risk	**Likelihood**	**Impact**	**Mitigation**
Scope creep	Medium	Medium	Define features clearly in the SRS and stick to MVP
Time overruns	High	Medium	Break work into small sprints and track velocity
UI complexity	Medium	Medium	Use simple, reusable HTML/CSS components first
Lack of real-time collaboration	Low	Low	SignalR can be added later as an enhancement
Data loss or corruption	Low	High	Implement backups and validation checks
Resource Allocation			
Resource	**Role**		

Lead Developer (Me)	Project Manager, Full-Stack Developer
Collaborators	None yet
Tools Used	Visual Studio, SQL Server Management Studio, GitHub
Dev/Test Environment	Localhost for development, IIS or Azure App Service for staging/deployment

2. Requirements Gathering and Analysis

Software Requirements Specification (SRS)

Description:
The Jira Software Clone is a full-stack web application designed to replicate the core functionality of Atlassian's Jira, a widely used issue and project tracking platform. This system allows software teams to manage projects, track progress, assign tasks, and collaborate effectively through visual workflows and real-time updates. The purpose of this project is to deepen full-stack development skills by architecting and implementing a scalable enterprise-grade solution from scratch.
Functional Requirements:

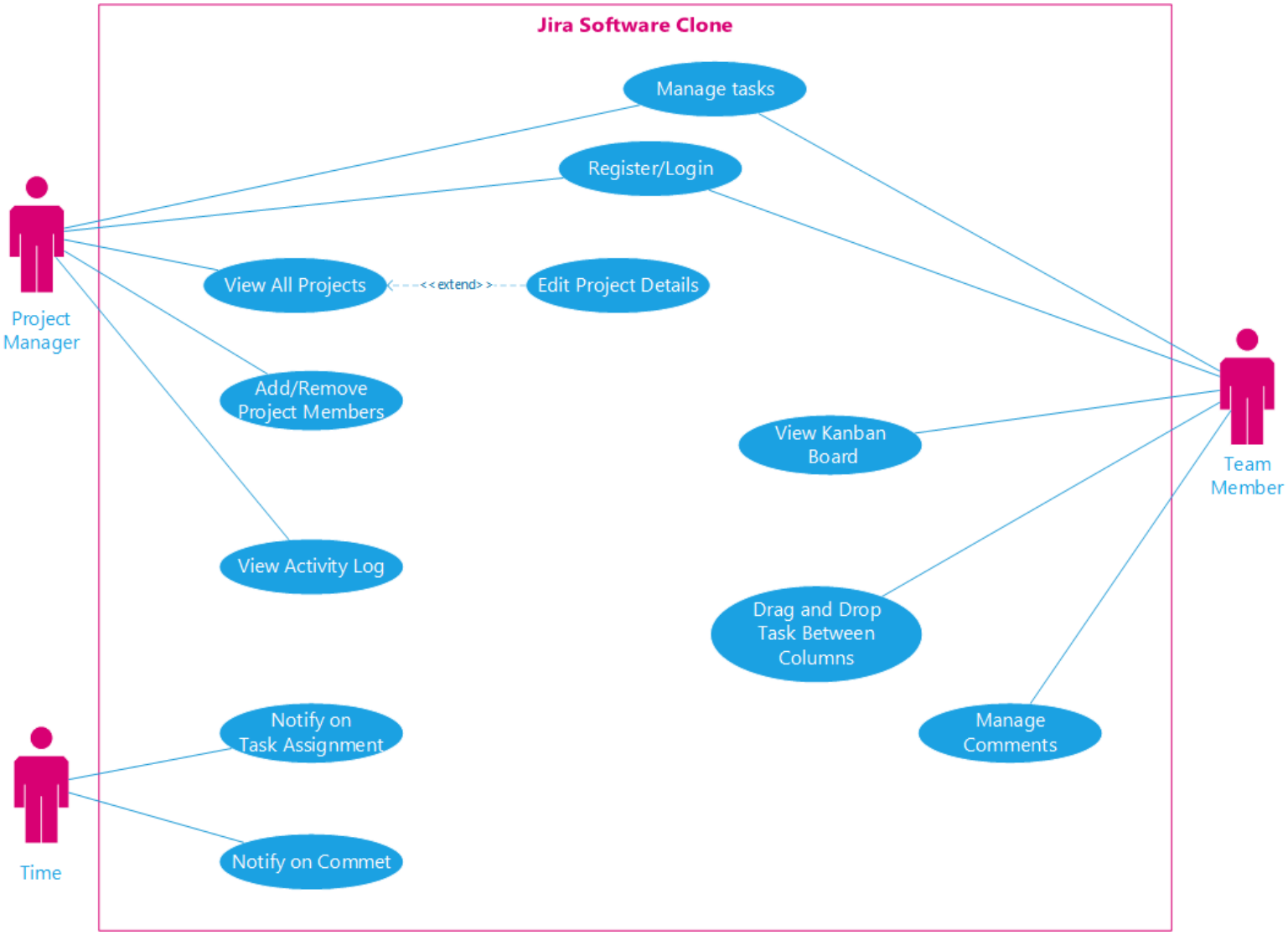
1. User Management	– User Registration – User Login & Logout – Role-based Access Control (Admin, Project Manager, Developer)
2. Project Management	– Create Project – Edit Project Details – Add/Remove Members to Project – View Project Overview
3. Task Management	– Create Task (within a project) – Assign Task to a User – Set Priority and Status – Add Descriptions and Deadlines – Edit/Delete Tasks – Drag-and-Drop Kanban Board
4. Task Status Workflow	– Create and Customize Task Statuses (To Do, In Progress, Done, etc.) – Move Tasks Between Statuses
5. Comment System	– Users can comment on tasks for discussions – Comments include author, timestamp, and text
6. Dashboard & Views	– Project Dashboard (List of Projects user is part of) – Task Board View (Kanban-style per project) – Task Detail View
7. Notifications (Planned)	– In-app notifications for task assignment and comments – Email notifications (future enhancement)
8. Activity Logging (Planned)	– Audit trail/log for task updates and user actions
9. RESTful API (Planned)	– Expose endpoints for integration with mobile clients or third-party apps
Non-Functional Requirements	
1. Usability	– UI must be intuitive and responsive across devices. – Accessible for users with basic technical knowledge.
2. Performance	– Should support up to 100 concurrent users without significant lag. – Task loading and board operations should respond in under 2 seconds.

3. Scalability	– System should support growth in the number of users, tasks, and projects. – Database design should accommodate future scaling.
4. Reliability and Availability	– Uptime target: 99.5% or higher (when deployed). – Should gracefully handle server failures (with retry logic where needed).
5. Security	– Passwords encrypted using industry-standard hashing (e.g., BCrypt). – Role-based access control to restrict feature access. – Input validation to prevent XSS, SQL injection, and CSRF attacks.
6. Maintainability	– Clean, modular code with comments and documentation. – Follows ASP.NET MVC architectural best practices. – Uses Entity Framework migrations for schema evolution.
7. Portability	– Web-based systems should run on any modern browser. – Deployment-ready on Windows Server with SQL Server.

Teck stack

Layer	Technology Used
Front-End	HTML5, CSS3, JavaScript
Back-End	ASP.NET MVC (C#)
ORM	Entity Framework
Database	SQL Server
Authentication	ASP.NET Identity
Future APIs	RESTful Web APIs

Use Case Diagram:



3. System Design

Class Diagram:

Business Rules:

- **User**: Can be part of multiple projects and can be assigned to tasks.
- **Project**: Has many tasks and team members.
- **Task**: Assigned to a user, has a status (like "To Do"), and belongs to one project.
- **TaskStatus**: A simple lookup for statuses like "To Do", "In Progress", etc.
- **ProjectMember**: Handles the many-to-many relationship between users and projects with a specific role.
- **Comment**: Users can leave comments on tasks for communication