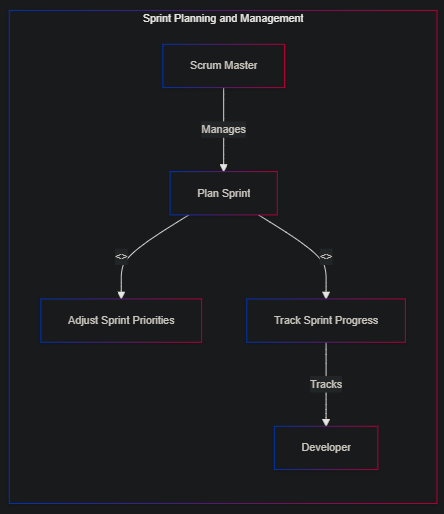
USE CASE DIAGRAMS

###### 1. Sprint Planning and Management

**Description**: Effectively manage sprint planning by organizing user stories, tasks, and bugs. Assign tasks to team members, track progress throughout the sprint, and adjust priorities as needed.

**Key Features**: Task boards, Gantt charts, sprint backlogs, progress tracking.

**Outcome**: Improved sprint execution with clear visibility into task assignments, ongoing progress, and the ability to respond quickly to changes in scope or priorities.

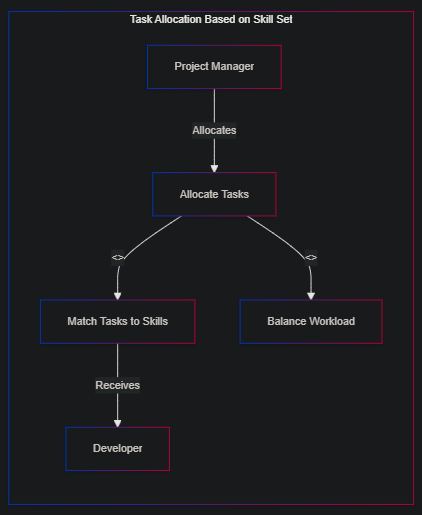


###### 2. Task Allocation Based on Skill Set

**Description**: Optimize resource utilization by automatically assigning tasks to developers based on their skill sets and availability. Analyze task requirements and match them with the most suitable team members.

**Key Features**: Skill matrices, task matching algorithms, workload balancing.

**Outcome**: Efficient use of team resources, ensuring that tasks are handled by the bestsuited developers, leading to faster and higherquality output.

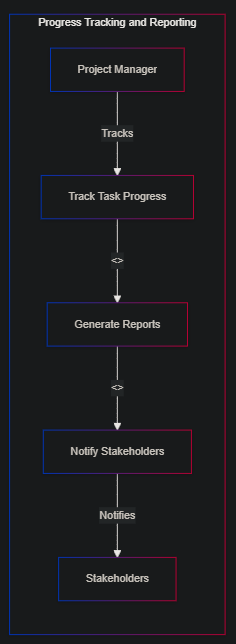


###### 3. Progress Tracking and Reporting

**Description**: Continuously monitor the progress of development tasks and milestones. Generate and share progress reports with stakeholders to keep everyone informed and aligned.

**Key Features**: Milestone tracking, real-time dashboards, automated reporting.

**Outcome**: Enhanced transparency and communication across teams and stakeholders, allowing for proactive management of timelines and resources.

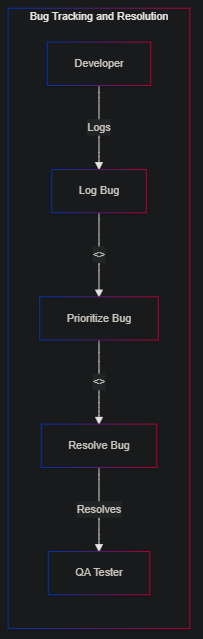


###### 4. Bug Tracking and Resolution

**Description**: Implement a system to capture, prioritize, and track bugs throughout the development process. Ensure that critical bugs are addressed promptly by setting up automated alerts and workflows.

**Key Features**: Bug tracking boards, priority tags, automated alerts.

**Outcome**: Faster bug resolution with clear tracking, leading to improved software quality and reduced production issues.

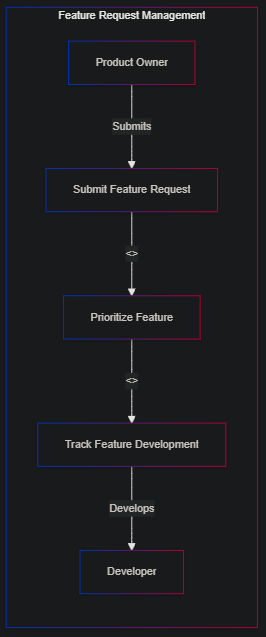


###### 5. Feature Request Management

**Description**: Manage feature requests from stakeholders by capturing them, prioritizing based on impact and feasibility, and tracking their development through to release.

**Key Features**: Request submission forms, prioritization frameworks, development tracking.

**Outcome**: Streamlined feature development process, ensuring that highimpact features are delivered efficiently and meet stakeholder expectations.

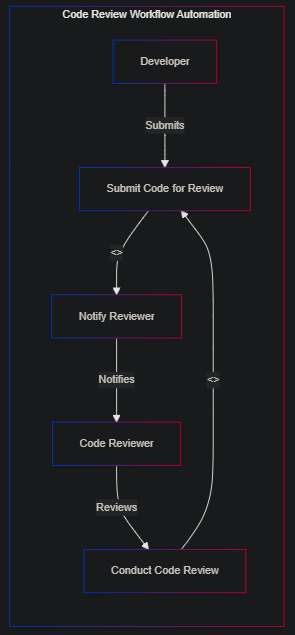


###### 6. Code Review Workflow Automation

**Description**: Automate the code review process by integrating with version control systems. Trigger notifications and task updates when code is ready for review, and track the review process to ensure timely feedback.

**Key Features**: Code review checklists, automated notifications, version control integration.

**Outcome**: Accelerated code review process, reducing bottlenecks and ensuring that quality checks are consistently applied before code is merged.

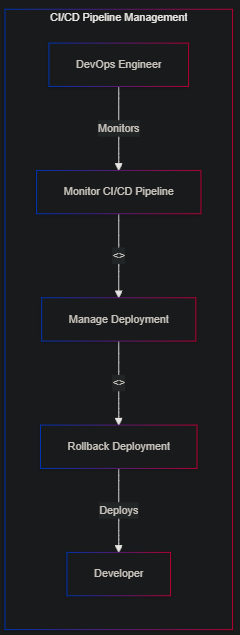


###### 7. Continuous Integration/Continuous Deployment (CI/CD) Pipeline Management

**Description**: Monitor and manage the CI/CD pipeline by tracking deployment status, managing rollbacks, and coordinating team responses to deployment issues.

**Key Features**: CI/CD pipeline dashboards, automated deployment tracking, rollback management.

**Outcome**: Improved coordination and visibility into the CI/CD process, reducing deployment risks and ensuring smooth, timely releases.

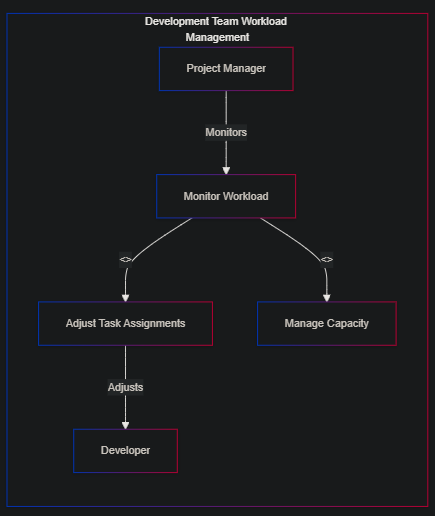


###### 8. Development Team Workload Management

**Description**: Monitor and manage the workload of development team members to ensure tasks are evenly distributed and prevent burnout. Adjust task assignments in realtime based on changing priorities or resource availability.

**Key Features**: Workload dashboards, realtime task reassignment, capacity planning tools.

**Outcome**: Balanced workload distribution, leading to sustained productivity and improved team morale.

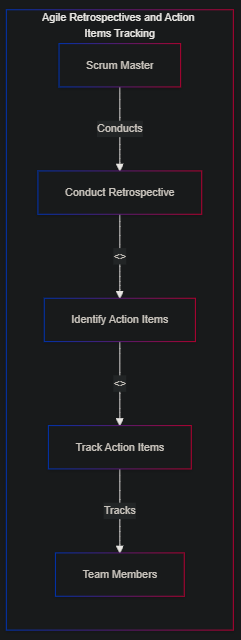


###### 9. Agile Retrospectives and Action Items Tracking

**Description**: Document and track action items from Agile retrospectives. Assign tasks to team members and monitor progress on implementing improvements identified during retrospectives.

**Key Features**: Retrospective boards, action item tracking, followup reminders.

**Outcome**: Continuous improvement in development practices, with a structured approach to implementing and tracking the outcomes of retrospectives.



###### 10. Release Planning and Execution

**Description**: Plan and execute software releases by managing tasks, dependencies, and timelines. Track progress and ensure that all release criteria are met before going live.

**Key Features**: Release calendars, dependency tracking, readiness checklists.

**Outcome**: More predictable and controlled release processes, reducing the risk of delays or issues at launch and ensuring alignment across all stakeholders.

