Introduction

Business Need

The South African software development industry, valued at USD 1.2 billion in 2023 and projected to reach USD 3.4 billion by 2030, is under increasing pressure to enhance team performance and deliver high-quality solutions efficiently. Engineering managers need data-driven insights to monitor and improve productivity, particularly through the adoption of industry-standard DORA metrics.

Project Scope

DevX360 is an AI-powered DevOps analytics platform designed to automatically track, analyse, and visualize key DORA metrics—**Deployment Frequency, Lead Time to Change, Mean Time to Recover**, and **Change Failure Rate**. By integrating with existing development tools, DevX360 provides actionable insights through real-time dashboards, automated reports, and AI-based recommendations to help engineering teams optimize workflows and reach elite performance levels.

User Stories

1. User Authentication

- **US1:** As a new user, I want to register via email/password or third-party providers (GitHub, Google) so that I can access the system securely.
- **US2:** As a security-conscious user, I want to enable Multi-Factor Authentication (MFA) during login so that my account remains protected.
- **US2.1:** As a security-conscious user, I want to be automatically logged out after 30 minutes of inactivity so that my session isn’t left vulnerable.
- **US3:** As a manager, I want to generate invite-only sign-up links for team members to ensure controlled access to the platform.

2. User Roles

- **US4:** As a manager, I want to view team performance metrics and detailed reports so that I can identify bottlenecks and optimize workflows.
- **US5:** As a team member, I want to view my personal metrics and receive AI-driven improvement suggestions so that I can enhance my contributions and commits.

3. DORA Metrics Collection

- **US6:** As a DevOps engineer, I want the system to automatically collect deployment frequency and lead time from GitHub and CI/CD tools so that I can monitor delivery performance without manual effort.
- **US7:** As a team lead, I want real-time updates on Mean Time to Recover (MTTR) so that I can address incidents promptly.

4. Team & Individual Metrics

- **US8:** As a manager, I want to view team-level metrics while respecting privacy so that I can foster collaboration without encouraging harmful competition.
- **US9:** As a developer, I want to see my individual contribution metrics privately so that I can self-assess and improve.

5. AI Features

- **US10:** As a developer, I want AI to analyze my pull requests and suggest code quality improvements so that I can reduce technical debt.
- **US11:** As a team member, I want AI-generated commit messages to standardize my workflow so that commit logs are clear and consistent.

6. Compliance & Tracking

- **US12:** As a manager, I want alerts when no commits are made on a workday so that I can ensure consistent progress and avoid code loss.
- **US13:** As a developer, I want reminders to commit code daily so that I stay aligned with team expectations.

7. Dashboard & Reporting

- **US14:** As a user, I want a role-based dashboard showing real-time DORA metrics so that I can track performance at a glance.
- **US15:** As a manager, I want automated weekly reports emailed to stakeholders so that I can share progress without manual effort.

8. Integrations

- **US16:** As a developer, I want seamless integration with GitHub and Jira so that data flows automatically into DevX360 without duplication.
- **US17:** As an admin, I want API access to connect custom tools so that the system adapts to our existing workflow.

9. Security & Compliance

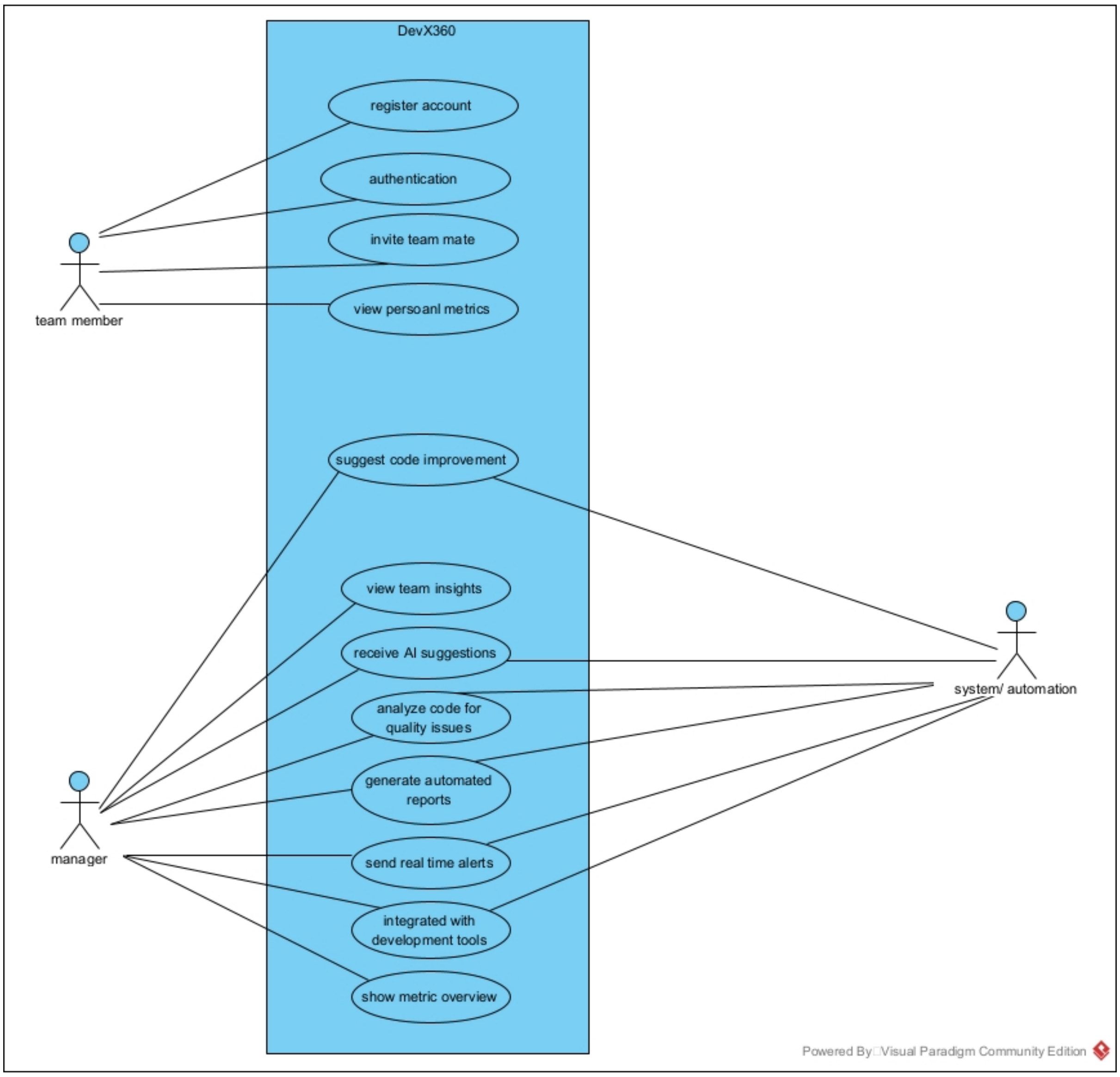
- **US18:** As a user, I want RBAC to restrict access to sensitive data so that only authorized roles view specific metrics.

- **US19:** As a compliance officer, I want GDPR-compliant data handling so that user privacy is legally protected.

10. Non-Functional Needs

- **US20:** As a user, I want the dashboard to load within 2 seconds so that I can work efficiently without delays.
- **US21:** As a global team, I want the system to support 10,000 concurrent users so that scalability is ensured during peak usage.

Use Case



Functional Requirements

FR1: User Authentication

- **FR1.1**: The system shall support user registration via email/password and third-party providers (e.g., GitHub, Google).
- **FR1.2**: The system shall implement Multi-Factor Authentication (MFA) for enhanced security.
- **FR1.3**: The system shall automatically log out users after 30 minutes of inactivity.
- **FR1.4**: The system shall support invite-only registration to ensure controlled team access.

FR2: User Profile Management

- **FR2.1**: The system shall provide a user profile page displaying user details including name, email, role, and join date.
- **FR2.2**: The system shall allow users to upload and update profile pictures/avatars.
- **FR2.3**: The system shall support JPEG, PNG, and GIF formats for profile pictures.
- **FR2.4**: The system shall automatically resize and optimize uploaded profile images for performance.

FR3: User Role Management

- **FR3.1**: The system shall implement Role-Based Access Control (RBAC) for "Manager" and "Team Member" roles.
- **FR3.2**: The system shall restrict access to data based on user roles and privacy settings.
- **FR3.3**: The system shall allow managers to invite team members and assign appropriate roles.

FR4: DORA Metrics Collection

- **FR4.1**: The system shall automatically collect **Deployment Frequency** metrics from CI/CD tools.
- **FR4.2**: The system shall calculate **Lead Time to Change** using data from version control systems.
- **FR4.3**: The system shall track **Mean Time to Recover (MTTR)** from incident and recovery data.
- **FR4.4**: The system shall compute **Change Failure Rate (CFR)** from deployment and rollback information.
- **FR4.5**: The system shall update DORA metrics within 30 seconds of data ingestion.

FR5: Team and Individual Metrics

- **FR5.1**: The system shall display team-level performance metrics accessible to managers.
- **FR5.2**: The system shall provide individual performance metrics with configurable privacy controls.
- **FR5.3**: The system shall support filtering of metrics by time periods and individual team members.

FR6: AI Code Analysis

- **FR6.1**: The system shall analyze pull requests to detect code quality issues.
- **FR6.2**: The system shall provide actionable improvement suggestions based on code analysis.
- **FR6.3**: The system shall identify patterns that may negatively affect DORA metrics performance.

FR7: AI-Assisted Commit Messages

- **FR7.1**: The system shall suggest standardized commit messages for developers.
- **FR7.2**: The system shall improve the consistency and clarity of the commit log.

FR8: Commit Compliance Tracking

- **FR8.1**: The system shall track daily commit activity per developer.
- **FR8.2**: The system shall generate alerts when no commits are made on designated workdays.
- **FR8.3**: The system shall send daily commit reminders to team members.

FR9: Performance Dashboard

- **FR9.1**: The system shall provide role-based dashboards showing real-time DORA metrics.
- **FR9.2**: The system shall allow users to customize their dashboard views.
- **FR9.3**: The system shall support drill-down features for in-depth performance analysis.

FR10: Automated Reporting

- **FR10.1**: The system shall generate automated weekly performance reports for stakeholders.
- **FR10.2**: The system shall send real-time alerts when metric thresholds are breached.
- **FR10.3**: The system shall allow users to schedule and configure custom reports.

FR11: Tool Integrations

- **FR11.1**: The system shall integrate with GitHub to collect repository and commit data.

- **FR11.2**: The system shall support integration with Jira for project and issue tracking.
- **FR11.3**: The system shall provide an open API for integration with custom or third-party tools.

Service Contracts

Authentication Service

Aspect	Description
Purpose	Manages user registration, login, and session management
Inputs	User credentials, authentication tokens, session data
Outputs	Authentication status, user roles, session tokens
Interface	REST API endpoints: - POST /auth/register - POST /auth/login - POST /auth/logout - GET /auth/verify

Metrics Collection Service

Aspect	Description
Purpose	Collects and processes DORA metrics from integrated tools
Inputs	GitHub webhooks, CI/CD pipeline data, deployment logs
Outputs	Calculated DORA metrics, processed performance data
Interface	Event-driven processing with REST API: - GET /metrics/team - GET /metrics/individual

AI Analysis Service

Aspect	Description
Purpose	Analyzes code quality and generates improvement suggestions

Inputs	Pull request data, commit information, code changes
Outputs	Code quality scores, improvement recommendations, pattern analysis
Interface	Internal service API: - POST /ai/analyze-pr - GET /ai/suggestions

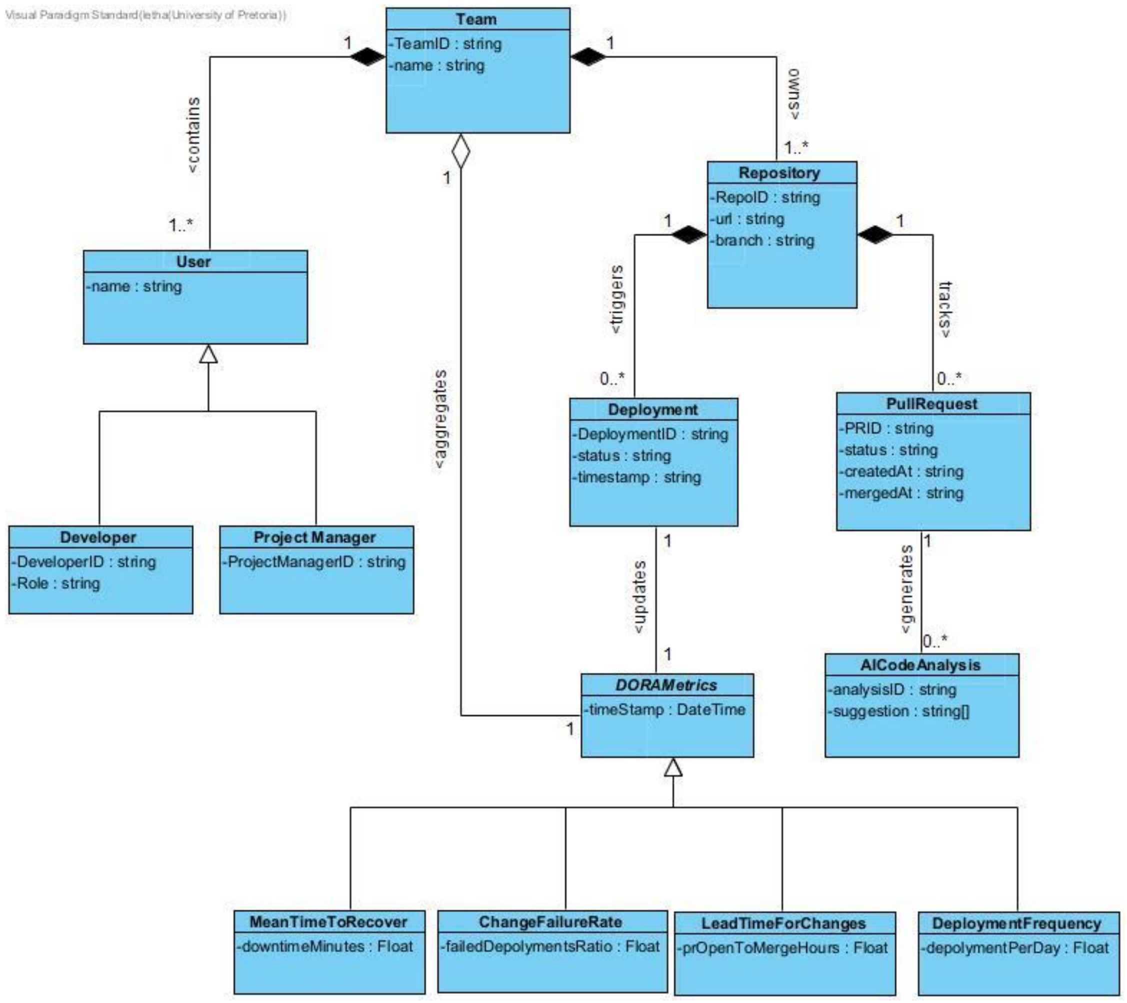
Dashboard Service

Aspect	**Description**
Purpose	Provides role-based dashboard data and visualizations
Inputs	User role, time filters, team selection
Outputs	Dashboard data, charts, performance summaries
Interface	REST API: - GET /dashboard/manager - GET /dashboard/member

Notification Service

Aspect	**Description**
Purpose	Manages alerts, reports, and user notifications
Inputs	Metric thresholds, user preferences, schedule configurations
Outputs	Email reports, in-app notifications, alert messages
Interface	Internal service: - POST /notifications/send - GET /notifications/preferences

Domain Model



Architectural Requirements

Quality Requirements

1. Performance Requirements

1.1. Dashboard Load Time: All dashboard pages must load within 2 seconds under normal network conditions.

1.2. Data Processing: DORA metrics calculations must be completed within 5 seconds of data ingestion from integrated tools.

1.3. API Response: API endpoints must respond within 500ms for 95% of requests under typical load.

2. Scalability Requirements

2.1. User Capacity: Support up to 10,000 concurrent users with no degradation in performance.

2.2. Repository Handling: Process data from 500 concurrent repositories simultaneously.

3. Reliability & Availability

3.1. Uptime: Achieve 99.9% annual uptime, excluding scheduled maintenance.

3.2. Real-Time Updates: Ensure metric updates reflect in the dashboard with a maximum latency of 30 seconds after data ingestion.

4. Security Requirements

4.1. Access Control: Implement role-based access control (RBAC) for Managers and Team Members.

4.2. Audits: Conduct security audits and vulnerability assessments.

5. Usability & Accessibility

5.1. Accessibility Compliance: Ensure the dashboard meets WCAG 2.1 AA accessibility standards.

5.2. Responsive Design: Optimize the interface for seamless use on desktop and mobile devices.

6. Compatibility & Integration

6.1. Tool Integration: Support integration with GitHub, GitLab, and Jira via their latest public APIs (2025 versions).

6.2. Browser Support: Ensure compatibility with Chrome, Firefox, Safari, and Edge (latest two versions).

7. Maintainability & Support

7.1. Microservices Updates: Allow independent deployment of microservices with ≤1 hour downtime per component.

7.2. Documentation: Provide detailed API documentation and architectural guidelines.

8. Legal & Compliance

8.1. Data Privacy: Comply with GDPR and POPIA regulations for user data protection.

8.2. User Consent: Obtain explicit user consent for data collection and processing during sign-up.

Architectural Patterns

- **Microservices Architecture:** Design the system using loosely coupled services for independent scaling and deployment.
- **API-First Design:** Ensure all integrations and internal components expose well-documented APIs for extensibility.
- **Event-Driven Processing:** Use asynchronous event handling for real-time metric updates and alerts.
- **Scalable Data Storage:** Employ cloud-native databases (e.g., Firebase, Supabase) to handle growing data volumes.

Design Patterns

Pattern	**Purpose**
Singleton Pattern	To ensure that the api makes one instance of the connection to the database
Observer Pattern	Enables real-time alerting and notification updates
Strategy Pattern	Allows multiple integration strategies for development tools

Factory Pattern	Facilitates creation of metric calculation services based on need

Constraints

- **GitHub Dependency:** The system must prioritize GitHub for sourcing metrics (e.g., commits, pull requests, CI/CD data).
- **Dashboard Functionality:** Dashboards must display at least 4 DORA metrics and support role-based privacy controls.
- **Hosting Limitations:** Deploy only on approved platforms (Vercel, Supabase, Firebase) to ensure cost efficiency.
- **Real-Time Data:** Metrics must reflect updates within 30 seconds of data ingestion.
- **Compliance Deadlines:** Adhere to GDPR and POPIA requirements from initial deployment.

Technology Requirements

Frontend Technologies

Component	**Technology**
Framework	JavaScript
Styling	HTML5, CSS3, Tailwind CSS
Data Visualization	D3.js for advanced charts and metric visualizations

Backend Technologies (assumption not implemented yet)

Component	**Technology**
Runtime	Node.js with Express framework
Database	Supabase (PostgreSQL with real-time capabilities)
AI Processing	Ollama (local LLM-based code analysis)

Integration Technologies

Component	**Technology**
Version Control	GitHub API
CI/CD Pipeline	GitHub Actions
Hosting	Vercel for frontend and serverless API hosting

Development Tools

Purpose	**Tool**
Code Quality	ESLint, Prettier
Testing	Jest (unit testing)
Monitoring	Built-in analytics and error tracking