Project: Defect Density and Code Quality Analysis

This report presents a comprehensive analysis of the project's software quality based on **Defect Density** and related metrics. By examining trends and breakdowns across different modules and development teams, we have identified key areas for improvement. The analysis is based on a simulated dataset spanning six months, providing a realistic overview of the codebase's health.

Our findings indicate a positive, downward trend in overall defect density, suggesting that recent quality assurance efforts are effective. However, the data also highlights that the **Authentication** module is a significant area of concern due to its high defect density, and **Team B** may benefit from additional support. This report provides actionable recommendations to build upon the positive trends and address existing vulnerabilities.

---

1. Project Overview and Methodology

The objective of this project was to create a dashboard to track code quality and identify areas that require attention. We used two primary datasets to achieve this:

- **Bugs Data**: A simulated record of 300 bugs, including key details such as module, team, and severity.
- **Lines of Code (LOC) Data**: A record of the codebase size for each module, tracked on a monthly basis.

The core metric used for this analysis is **Defect Density**, calculated as:

Defect Density=Total Lines of CodeTotal Number of Bugs×1000

These datasets were processed and analyzed using **Tableau Public** to generate interactive visualizations that form the basis of our findings.

---

2. Analysis of Key Findings

The visualizations from the dashboard reveal several important trends and insights.

2.1 Overall Quality Trend

The line chart below shows a clear and encouraging **downward trend** in overall defect density from January to June 2025. This suggests that the current quality control measures—such as code reviews, unit testing, and developer training—are having a positive impact.

2.2 Defect Breakdown by Module

This analysis pinpoints the most problematic parts of the codebase. The **Authentication** module consistently shows a significantly higher defect density compared to all other modules. This indicates that this specific area of the application is a primary source of bugs and requires immediate attention.

2.3 Defect Breakdown by Team

Examining defect density by team helps in identifying where additional support may be needed. While the bug count is well-distributed, **Team B** shows a slightly higher average defect density than the other teams. This could be due to a variety of factors, such as working on a more complex module or a recent technology, and it merits further investigation.

2.4 Bug Severity Distribution

The pie chart below provides context to the number of bugs by showing their impact. The majority of bugs found are of **Medium** or **High** severity, which highlights a risk to the application's stability. While the total number of bugs is decreasing, a focus on reducing these high-impact issues is critical.

---

3. Recommendations

Based on the data and analysis, the following recommendations are proposed to further enhance code quality and maintain the positive momentum:

1. **Prioritize Refactoring**: A dedicated refactoring sprint for the **Authentication** module is highly recommended to reduce technical debt and improve code quality in this critical area.
2. **Targeted Training**: Offer a workshop or peer-mentoring session for **Team B** focused on advanced testing techniques and code review best practices.
3. **Implement Continuous Monitoring**: Integrate the dashboard into the regular project management process to continuously monitor key metrics after each release and proactively address any quality degradation.

These steps will help ensure that the team not only maintains but accelerates the current positive trend in software quality.