UTT
UNIVERSIDAD TECNOLÓGICA DE TIJUANA
GOBIERNO DE BAJA CALIFORNIA

TEMA: Strategy versioning.

PRESENTADO POR: Ramírez Aispuro Juan José

GRUPO: 10-B

MATERIA: Desarrollo móvil integral

PROFESOR: Ray Brunett Parra Galaviz

Tijuana, Baja California, 01/07/2024

Recommended Versioning Strategy: **Semantic Versioning (SemVer)**

Why Choose Semantic Versioning?

1. Clear and Predictable Structure:

Semantic Versioning uses a format of MAJOR.MINOR.PATCH (e.g., 1.2.3), where each segment has a specific meaning:

- **MAJOR:** Increments for backward-incompatible changes.
- **MINOR:** Increments for backward-compatible new features.
- **PATCH:** Increments for backward-compatible bug fixes.

This clarity helps developers and users quickly understand the impact of updates.

2. Facilitates Dependency Management:

With a consistent versioning system, libraries and applications can specify compatibility ranges. For example, a dependency might require >=1.2.0, <2.0.0, ensuring it works with the latest non-breaking updates.

3. Enhances Communication:

Semantic Versioning clearly communicates the nature of changes in a release, reducing confusion for teams, collaborators, and end-users. This transparency is critical for maintaining trust in widely-used software.

4. Standardized Approach:

SemVer is widely adopted across industries and ecosystems, making it a familiar and standardized approach that aligns with tools like package managers (e.g., npm, pip).

Semantic Versioning is ideal for projects where multiple stakeholders rely on the software and need clear guidance about update impacts. It is especially useful for public APIs, libraries, or frameworks that require compatibility and long-term maintenance.