

Release Planning

Emmanuel BRUNO
Université de Toulon, LIS UMR CNRS 7020
emmanuel.bruno@univ-tln.fr

2026-10-14

Table des matières

1	Release Planning	1
1.1	Release Philosophy	1
1.2	Release Cadence	1
1.3	Release Types	1
1.3.a	Major Releases (x.0.0)	1
1.3.b	Minor Releases (0.x.0)	2
1.3.c	Patch Releases (0.0.x)	2
1.4	Release Planning Timeline	2
1.4.a	Major Release Timeline (12 weeks)	2
1.4.b	Minor Release Timeline (4 weeks)	2
1.4.c	Patch Release Timeline (1 week)	2
1.5	Release Planning Process	2
1.5.a	1. Roadmap Planning	2
1.5.b	2. Release Scope Definition	2
1.5.c	3. Sprint Planning	2
1.5.d	4. Release Preparation	2
1.5.e	5. Release Validation	2
1.5.f	6. Release Deployment	2
1.5.g	7. Post-Release Activities	2
1.6	Release Artifacts	3
1.7	Feature Tracking	3
1.8	Example Release Plan Template	3
1.9	Stakeholder Communication	4
1.10	Risk Management in Releases	4
1.10.a	Common Release Risks	4
1.10.b	Contingency Planning	4
1.11	Feature Flagging Strategy	4

1.11.a	Feature Flag Lifecycle	4
1.12	Continuous Delivery Pipeline	4
1.13	Documentation Requirements	5
1.14	Version Control Strategy	5
1.15	Post-Release Evaluation	5

1 Release Planning

This document outlines our approach to planning and scheduling releases for the Fitness Tracker application.

1.1 Release Philosophy

Our release strategy follows these principles:

1. **Feature-driven releases** with time-boxed development cycles
2. **Progressive rollout** to minimize risk
3. **Regular cadence** for predictability
4. **Continuous delivery** of value to users

1.2 Release Cadence

- **Major Releases:** Quarterly (4 per year)
- **Minor Releases:** Monthly
- **Patch Releases:** As needed for critical fixes

1.3 Release Types

1.3.a Major Releases (x.0.0)

- Significant new features
- Major UI/UX changes
- Breaking changes to APIs
- Extensive testing required
- Marketing coordination needed

1.3.b Minor Releases (0.x.0)

- New features that don't significantly change core functionality
- Enhancements to existing features
- Non-breaking improvements
- Regular testing cycle

1.3.c Patch Releases (0.0.x)

- Bug fixes
- Security patches
- Performance improvements
- Minimal UI changes
- Expedited testing

1.4 Release Planning Timeline

1.4.a Major Release Timeline (12 weeks)

Week	Activities
1-2	Feature planning, roadmap refinement
3-8	Feature development (3 sprints)
9-10	Feature freeze, stabilization
11	Release candidate, beta testing
12	Final testing, release preparation

1.4.b Minor Release Timeline (4 weeks)

Week	Activities
1-2	Feature development
3	Feature freeze, testing
4	Final testing, release

1.4.c Patch Release Timeline (1 week)

Day	Activities
1-2	Fix development and code review
3-4	Testing and validation
5	Release

1.5 Release Planning Process

1.5.a 1. Roadmap Planning

- Quarterly roadmap review with stakeholders
- Align features with business objectives
- Prioritize features based on user value and strategic goals

1.5.b 2. Release Scope Definition

- Select features from prioritized backlog
- Define clear release goals
- Document features, improvements, and fixes
- Obtain stakeholder approval on scope

1.5.c 3. Sprint Planning

- Break down release scope into sprint-sized increments
- Allocate features to sprints
- Define dependencies and critical path

1.5.d 4. Release Preparation

- Create release branch when feature development completes
- Conduct code freeze for stabilization
- Perform regression testing
- Update release documentation
- Prepare marketing materials (if applicable)

1.5.e 5. Release Validation

- Alpha testing with internal team
- Beta testing with selected users
- Performance testing
- Security review
- Final approval from product owner

1.5.f 6. Release Deployment

- Staged rollout (if applicable)
- Monitoring for issues
- Ready rollback plan if needed

1.5.g 7. Post-Release Activities

- Monitor key metrics and user feedback
- Document lessons learned
- Plan post-release fixes if needed

1.6 Release Artifacts

For each release, the following artifacts should be prepared:

- **Release Notes:** User-facing documentation of changes
- **Technical Documentation:** Updates to developer documentation
- **Deployment Instructions:** Steps for deploying the release
- **Rollback Plan:** Procedure in case issues are detected
- **Test Results:** Summary of testing outcomes

1.7 Feature Tracking

Features for each release are tracked in JIRA with the following information:

- Feature ID and name
- Feature description and acceptance criteria
- Target release version
- Current status
- Dependencies
- Assigned team members
- Testing requirements

1.8 Example Release Plan Template

```
# Release Plan: v1.2.0
```

```
## Release Information
```

- **Version:** 1.2.0
- **Release Date:** June 15, 2023
- **Release Manager:** [Name]

```
## Release Goals
```

- Introduce workout sharing functionality
- Improve workout analytics dashboard
- Fix reported issues from v1.1.0

```
## Features and Enhancements
```

1. **Workout Sharing** (FEAT-123)

- Share workouts via social media
- Export workout data as PDF
- Send workout invitations to friends

```
2. Analytics Dashboard Improvements (FEAT-124)
```

- Weekly progress charts
- Improved data visualization
- Workout comparison tool

```
3. Bug Fixes
```

- Fix crash when creating custom exercises (BUG-456)
- Fix calorie calculation accuracy (BUG-457)
- Fix synchronization issues with cloud backup (BUG-458)

```
## Dependencies
```

- Backend API changes for workout sharing
- Design assets for new analytics visualizations

```
## Testing Plan
```

- Unit testing: All new features
- Integration testing: Sharing functionality
- Performance testing: Analytics dashboard
- Beta testing: 1 week with selected users

```
## Rollout Plan
```

- Internal testing: June 1-5
- Beta release: June 6-12
- Production release: June 15
- Staged rollout: 10% → 50% → 100% over 3 days

```
## Success Metrics
```

- 20% of users share at least one

workout

- Increased engagement with analytics dashboard
- Reduction in reported crashes by 50%

1.9 Stakeholder Communication

- **Pre-Release:** Share release plan and expected features
- **During Development:** Weekly status updates on feature completion
- **Pre-Launch Testing:** Detailed test results and identified issues
- **Release Announcements:** Staggered communications to different user segments
- **Post-Release:** Gather feedback and communicate resolution of any issues

1.10 Risk Management in Releases

1.10.a Common Release Risks

Risk	Mitigation Strategy
Feature delays	Buffer time in schedule; feature toggles for incomplete work
Post-release bugs	Comprehensive testing; staged rollout; rollback plan
User adoption issues	Beta testing with target users; clear documentation
Infrastructure problems	Load testing; capacity planning; redundant systems
Store approval delays	Early submission; follow platform guidelines strictly

1.10.b Contingency Planning

For each release, establish: - Go/No-Go criteria for launch decisions - Emergency fix protocol for critical issues - Communication templates for delay announcements - Responsibility matrix for different types of incidents

1.11 Feature Flagging Strategy

Feature flags allow for greater control and safer releases:

- **Development Flags:** Enable features still under development
- **Release Flags:** Control gradual rollout of completed features
- **Experiment Flags:** A/B testing different implementations
- **Ops Flags:** Emergency toggles for problematic features

1.11.a Feature Flag Lifecycle

1. Create flag with clear naming convention
2. Implement feature behind flag
3. Test both flag states thoroughly
4. Deploy with flag in appropriate initial state
5. Adjust flag state based on metrics/feedback
6. Remove flag when feature is stable

1.12 Continuous Delivery Pipeline

Our release process is supported by a robust CI/CD pipeline:

1. **Build Phase:**
 - Compile code and run static analysis
 - Generate build artifacts
2. **Test Phase:**
 - Run unit and integration tests
 - Perform automated UI testing
3. **Quality Gate:**
 - Code coverage verification
 - Performance benchmarking
4. **Deployment Phase:**
 - Deploy to test environment
 - Deploy to staging environment

- Deploy to production (manual approval)
5. **Monitoring Phase:**
 - Track adoption metrics
 - Monitor for increased error rates
 - Collect user feedback

1.13 Documentation Requirements

Each release must include updates to:

1. **User-Facing Documentation:**
 - Feature guides
 - FAQ updates
 - Tutorial videos (for major features)
2. **Internal Documentation:**
 - Architecture changes
 - API documentation
 - Known issues and workarounds
3. **Release Notes:**
 - Clear feature descriptions
 - Bug fixes
 - Known limitations
 - Upgrade instructions

1.14 Version Control Strategy

Our release branches follow this strategy:

- `main` - Production-ready code
- `develop` - Integration branch for features
- `release/vX.Y.Z` - Release preparation branches
- `feature/name` - Individual feature development
- `hotfix/name` - Emergency fixes for production

1.15 Post-Release Evaluation

After each release, we conduct:

1. **Success Metrics Analysis:**
 - Feature adoption rate
 - User engagement changes
 - Error rate comparison
 - Performance impact
2. **Retrospective Meeting:**

- What went well
 - What could be improved
 - Action items for next release
3. **Technical Debt Assessment:**
 - Identify shortcuts taken for release
 - Schedule remediation work

This comprehensive approach to release planning ensures we deliver value to users consistently while maintaining a high-quality application.