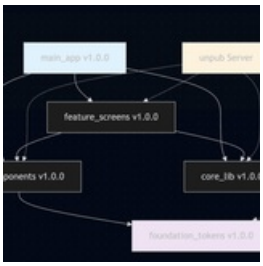
Flutter Design System - Multi Module Architecture

🎯 Overview

Proposal untuk membangun Design System Flutter menggunakan arsitektur multi-module dengan Melos sebagai monorepo tool dan Unpub untuk private package management.

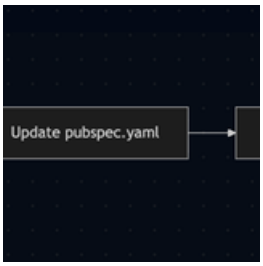
🏗️ Architecture Diagram

Module Dependency Structure



Development Workflow

Package Publishing Flow



📦 Module Structure

1. Foundation Tokens (`foundation_tokens`)

Purpose: Base design system primitives

```
Version: 1.0.0
Dependencies: None
Contents:
  - Color palette & tokens
  - Typography scales
  - Spacing constants
  - Border radius values
  - Shadow definitions
  - Theme configurations
```

2. UI Components (`ui_components`)

Purpose: Reusable UI building blocks

```
Version: 1.0.0
Dependencies: foundation_tokens ^1.0.0
Contents:
  Atomic:
    - CustomText, CustomButton
    - CustomCard, CustomIcon
  Compound:
    - SearchBar, AppBar
    - LoadingButton, ImageWithLoading
  Specialized:
    - SuccessButton, WarningAlert
    - ErrorDialog, ProgressIndicator
```

3. Feature Screens (`feature_screens`)

Purpose: Pre-built common screens

```
Version: 1.0.0
Dependencies:
  - ui_components ^1.0.0
  - core_lib ^1.0.0
Contents:
  - LoginScreen, RegisterScreen
  - OnboardingFlow, SplashScreen
  - ProfileScreen, SettingsScreen
  - ErrorScreen, MaintenanceScreen
```

4. Core Libraries (`core_lib`)

Purpose: Business logic utilities

```
Version: 1.0.0
Dependencies: None
Contents:
  - HTTP client & interceptors
  - Local storage wrapper
  - JSON serialization helpers
  - Utility functions
  - Error handling
  - Logging system
```

5. Main App (`main_app`)

Purpose: Final application

```
Version: 1.0.0
Dependencies:
  - feature_screens ^1.0.0
  - ui_components ^1.0.0
  - core_lib ^1.0.0
Contents:
  - App-specific screens
  - Routing configuration
  - App initialization
```

Technical Implementation

Melos Configuration

```
# melos.yaml
name: flutter_design_system
repository: https://github.com/company/flutter-design-system

packages:
  - packages/**
  - apps/**

scripts:
  analyze: flutter analyze
  test: flutter test
  build: flutter build apk
  pub:get: flutter pub get
```

Unpub Setup

```
# unpub.yaml
database:
  url: 'mongodb://localhost:27017/unpub'
storage:
  type: 'file'
  path: './unpub-storage'
upstream:
  type: 'pub'
  url: 'https://pub.dev'
```

Folder Structure

```
flutter_design_system/
├── melos.yaml
├── packages/
│   ├── foundation_tokens/
│   ├── ui_components/
│   ├── feature_screens/
│   └── core_lib/
├── apps/
│   ├── main_app/
│   └── example_app/
└── tools/
    ├── unpub_config/
    └── scripts/
```

Benefits & Advantages

For Development Team

- **Modularity**: Clear separation of concerns
- **Parallel Development**: Teams can work on different modules simultaneously
- **Code Reusability**: Components shared across multiple apps
- **Version Control**: Independent versioning per module
- **Testing**: Isolated unit testing per module

For Business

- **Faster Development**: Reuse existing components
- **Consistency**: Unified design language across apps
- **Maintainability**: Centralized design system updates
- **Scalability**: Easy to add new apps using existing modules

Technical Benefits

- **Dependency Management**: Clear dependency hierarchy
- **Build Optimization**: Only rebuild changed modules
- **Private Packages**: Secure internal package distribution
- **Automated Versioning**: Consistent release management

⚠️ Risk Mitigation

Technical Risks

- **Dependency Hell**: Strict semver policies & automated testing
- **Breaking Changes**: Deprecation warnings & migration guides
- **Package Size**: Tree-shaking & modular imports

Team Risks

- **Learning Curve**: Comprehensive documentation & training
- **Coordination**: Regular sync meetings & clear ownership
- **Resistance to Change**: Gradual migration & clear benefits demonstration

🎯 Recommendations

1. **Start Small**: Begin with foundation_tokens and core components
2. **Automate Everything**: Use Melos scripts for common tasks
3. **Document Thoroughly**: Include examples and migration guides
4. **Test Extensively**: Unit, integration, and visual regression tests
5. **Monitor Usage**: Track which components are used most frequently

👥 Team Responsibilities

Frontend Team Lead

- Architecture oversight & code reviews
- Module design approval
- Cross-team coordination

Senior Developers

- Module implementation
- Testing strategy
- Documentation creation

Junior Developers

- Component development
- Test case writing
- Documentation updates

DevOps Team

- Unpub server maintenance
- CI/CD pipeline setup
- Release automation
