# FanFanLokMapper Implementation Order

## Phase 1: Foundation & Data Models (No Dependencies)

1. **Constants.kt** - App constants and configuration values
2. **Logger.kt** - Debug logging utility
3. **CardPosition.kt** - Simple data class for coordinates
4. **DetectionResult.kt** - Detection result wrapper

## Phase 2: Core Processing Logic

5. **ImageProcessor.kt** - Basic image loading and bitmap operations
6. **BorderDetector.kt** - Rectangle border detection algorithms
7. **GridMapper.kt** - 4x6 grid layout mapping logic
8. **FilterResultsUseCase.kt** - Size threshold filtering

## Phase 3: Domain Layer

9. **ImageRepositoryInterface.kt** - Repository interface definition
10. **DetectCardsUseCase.kt** - Card detection orchestration
11. **ProcessImageUseCase.kt** - Main processing pipeline

## Phase 4: Data Layer

12. **JsonExporter.kt** - JSON coordinate export functionality
13. **ImageRepository.kt** - Image repository implementation

## Phase 5: UI Components

14. **ImagePicker.kt** - File picker component
15. **CardOverlay.kt** - Green rectangle overlays with long-press
16. **DebugConsole.kt** - Debug logging display

## Phase 6: ViewModels & State Management

17. **ImageProcessingViewModel.kt** - Image processing state
18. **MainViewModel.kt** - Main screen state management

## Phase 7: Screens & Integration

19. **ImageProcessingScreen.kt** - Image display with overlays
20. **MainScreen.kt** - Main UI screen
21. **MainActivity.kt** - Update main activity integration

## Phase 8: Resources & Testing (Optional)

22. **ic_folder.xml** - File picker icon
23. **BorderDetectorTest.kt** - Unit tests for border detection
24. **GridMapperTest.kt** - Unit tests for grid mapping
25. **FilterResultsTest.kt** - Unit tests for filtering

## Implementation Strategy

### Why This Order?

- **Bottom-up approach**: Build foundational components first
- **Dependency management**: Each file only depends on previously implemented files
- **Early testing**: Core logic can be tested before UI implementation
- **Incremental progress**: Each phase produces working, testable components

### Validation Points

- **After Phase 2**: Test core image processing algorithms
- **After Phase 4**: Test complete processing pipeline with JSON output
- **After Phase 6**: Test UI state management
- **After Phase 7**: Full application integration testing

### Alternative Approach (Top-down)

If you prefer to see UI results quickly, we could start with:

1. Basic UI mockups (Phase 5-7)
2. Stub implementations for processing
3. Fill in real algorithms later

Which approach would you prefer to take?