# 🛒 간단한 쇼핑 앱 - 7일 완성 가이드

## \*\*Day 1: 상품 목록 기본 구조 + GridView 성능 최적화\*\*

\*\*복습 개념\*\*: GridView와 ListView 성능 최적화

\*\*구현 목표\*\*: 상품 목록 화면과 상세 화면의 기본 구조 완성

### \*\*🗓️ Day 1 진행 가이드\*\*

\*\*1️⃣ 오늘의 목표\*\*

- 상품 목록을 보여주는 GridView 구현

- 상품 상세 페이지 기본 구조

- 성능 최적화된 이미지 표시

\*\*2️⃣ 기존 코드 연결점\*\*

- 새 프로젝트 생성부터 시작

- main.dart에 MaterialApp 설정

- 상품 데이터 모델 클래스 생성

\*\*3️⃣ 구현 순서\*\*

1. 프로젝트 생성 및 기본 설정

2. Product 데이터 모델 클래스 생성

3. 메인 화면 (ProductListScreen) - GridView.builder 구현

4. 상품 상세 화면 (ProductDetailScreen) 기본 구조

5. GridView 성능 최적화 (cacheExtent, itemExtent 적용)

\*\*4️⃣ 주의사항\*\*

- GridView.builder 사용으로 메모리 효율성 확보

- 이미지는 placeholder 사용 (네트워크 이미지는 나중에)

- Null Safety 적용한 Product 모델 설계

\*\*5️⃣ 완성 기준\*\*

- 상품 목록 GridView 표시 (2열 구조)

- 상품 터치 시 상세 화면 이동

- 성능 최적화된 스크롤링

---

## \*\*Day 2: 상품 검색 시스템 + Form 검증\*\*

\*\*복습 개념\*\*: 복잡한 Form 검증과 TextField 최적화

\*\*구현 목표\*\*: 실시간 검색과 필터링 기능 구현

### \*\*🗓️ Day 2 진행 가이드\*\*

\*\*1️⃣ 오늘의 목표\*\*

- 상품 검색 기능 구현

- 카테고리별 필터링 시스템

- 검색어 유효성 검증

\*\*2️⃣ 기존 코드 연결점\*\*

- ProductListScreen에 검색 AppBar 추가

- 상품 목록에 필터링 로직 적용

- setState를 활용한 실시간 업데이트

\*\*3️⃣ 구현 순서\*\*

1. 검색 TextField와 검색 아이콘 추가

2. 카테고리 필터 Chip 위젯들 구성

3. 검색어 기반 상품 필터링 메서드 구현

4. 카테고리별 필터링 메서드 추가

5. 검색 결과 없을 때 Empty State 구현

\*\*4️⃣ 주의사항\*\*

- TextField의 debouncing으로 성능 최적화

- 검색어 trim() 처리와 null 체크

- 빈 검색어일 때 전체 목록 표시

\*\*5️⃣ 완성 기준\*\*

- 실시간 상품 검색 동작

- 카테고리별 필터링 기능

- 검색 결과 없을 때 적절한 UI 표시

---

## \*\*Day 3: 장바구니 상태 관리 + Provider 패턴\*\*

\*\*복습 개념\*\*: Provider를 활용한 복잡한 상태 관리

\*\*구현 목표\*\*: 장바구니 기능과 전역 상태 관리 시스템

### \*\*🗓️ Day 3 진행 가이드\*\*

\*\*1️⃣ 오늘의 목표\*\*

- Provider 패턴으로 장바구니 상태 관리

- 상품 추가/제거/수량 조절 기능

- 장바구니 화면 구현

\*\*2️⃣ 기존 코드 연결점\*\*

- pubspec.yaml에 provider 패키지 추가

- main.dart에 ChangeNotifierProvider 설정

- CartProvider 클래스 생성

\*\*3️⃣ 구현 순서\*\*

1. CartItem 데이터 모델 클래스 생성

2. CartProvider 클래스 구현 (ChangeNotifier 상속)

3. 상품 상세 화면에 장바구니 추가 버튼

4. 장바구니 화면 (CartScreen) 구현

5. Consumer 위젯으로 실시간 UI 업데이트

\*\*4️⃣ 주의사항\*\*

- Provider의 context 전달 주의 (listen: false 적절히 사용)

- 장바구니 아이템 중복 체크 로직

- notifyListeners() 호출 타이밍

\*\*5️⃣ 완성 기준\*\*

- 상품을 장바구니에 추가/제거 가능

- 수량 조절 기능 동작

- 장바구니 총 금액 실시간 계산

---

## \*\*Day 4: 좋아요 기능 + SharedPreferences 데이터 저장\*\*

\*\*복습 개념\*\*: SharedPreferences와 JSON 데이터 영속화

\*\*구현 목표\*\*: 좋아요 기능과 데이터 영구 저장

### \*\*🗓️ Day 4 진행 가이드\*\*

\*\*1️⃣ 오늘의 목표\*\*

- 상품 좋아요/찜 기능 구현

- SharedPreferences로 데이터 저장

- 앱 재시작 후에도 데이터 유지

\*\*2️⃣ 기존 코드 연결점\*\*

- shared\_preferences 패키지 추가

- FavoriteProvider 클래스 생성

- 상품 목록과 상세 화면에 하트 아이콘 추가

\*\*3️⃣ 구현 순서\*\*

1. FavoriteProvider 클래스 구현

2. SharedPreferences 저장/로드 메서드 작성

3. 상품 카드에 좋아요 버튼 추가

4. 좋아요 목록 화면 구현

5. JSON 인코딩/디코딩으로 복잡한 데이터 처리

\*\*4️⃣ 주의사항\*\*

- JSON 파싱 시 try-catch로 안전한 에러 처리

- 좋아요 상태 변경 시 즉시 저장

- 앱 시작 시 저장된 데이터 자동 로드

\*\*5️⃣ 완성 기준\*\*

- 좋아요 토글 기능 동작

- 앱 재시작 후 좋아요 상태 유지

- 좋아요한 상품만 따로 보는 화면

---

## \*\*Day 5: 주문 프로세스 + 복잡한 계산 로직\*\*

\*\*복습 개념\*\*: 복잡한 비즈니스 로직과 계산 처리

\*\*구현 목표\*\*: 주문 기능과 가격 계산 시스템

### \*\*🗓️ Day 5 진행 가이드\*\*

\*\*1️⃣ 오늘의 목표\*\*

- 주문하기 기능 구현

- 복잡한 가격 계산 (할인, 배송비, 세금)

- 주문 내역 관리

\*\*2️⃣ 기존 코드 연결점\*\*

- 장바구니 화면에 주문하기 버튼 추가

- OrderProvider 클래스 생성

- 주문 확인 화면 구현

\*\*3️⃣ 구현 순서\*\*

1. Order와 OrderItem 데이터 모델 생성

2. 가격 계산 로직 구현 (할인율, 배송비 계산)

3. 주문 확인 화면 구현

4. 주문 완료 처리 및 장바구니 초기화

5. 주문 내역 목록 화면

\*\*4️⃣ 주의사항\*\*

- 금액 계산 시 double 정밀도 문제 주의

- 주문 완료 후 상태 초기화 로직

- 주문 번호 생성 알고리즘

\*\*5️⃣ 완성 기준\*\*

- 장바구니에서 주문 진행 가능

- 정확한 총액 계산 (상품가격 + 배송비)

- 주문 완료 후 주문 내역 확인 가능

---

## \*\*Day 6: 이미지 처리 + 카테고리 필터 고도화\*\*

\*\*복습 개념\*\*: 이미지 최적화와 위젯 성능 향상

\*\*구현 목표\*\*: 앱 고도화 및 사용자 경험 개선

### \*\*🗓️ Day 6 진행 가이드\*\*

\*\*1️⃣ 오늘의 목표\*\*

- 상품 이미지 최적화 구현

- 고급 카테고리 필터링

- 앱 전체 성능 최적화

\*\*2️⃣ 기존 코드 연결점\*\*

- cached\_network\_image 패키지 추가

- 상품 이미지 교체 및 최적화

- 필터링 UI 개선

\*\*3️⃣ 구현 순서\*\*

1. CachedNetworkImage로 이미지 최적화

2. 이미지 로딩 중 Shimmer 효과 추가

3. 다중 카테고리 선택 필터 구현

4. 가격 범위 슬라이더 필터 추가

5. 정렬 옵션 (가격순, 인기순) 구현

\*\*4️⃣ 주의사항\*\*

- 이미지 캐싱으로 네트워크 사용량 최적화

- 필터 조합 시 성능 고려

- Shimmer 애니메이션 과도하지 않게 적용

\*\*5️⃣ 완성 기준\*\*

- 빠른 이미지 로딩과 캐싱

- 다양한 필터링 옵션 동작

- 부드러운 사용자 경험

---

## \*\*Day 7: UI 개선 + 사용자 가이드\*\*

\*\*목표\*\*: 프로젝트 마무리 및 학습 정리

\*\*작업\*\*: 코드 정리, 문서 작성, 학습 노트 정리

### \*\*🗓️ Day 7 진행 가이드\*\*

\*\*1️⃣ 오늘의 목표\*\*

- 전체 UI/UX 일관성 개선

- 에러 처리 및 예외 상황 대응

- 프로젝트 문서화

\*\*2️⃣ 작업 내용\*\*

1. 전체 앱 테마 일관성 확인

2. 로딩 상태와 에러 상태 UI 개선

3. 사용자 피드백 (SnackBar, 다이얼로그) 추가

4. 코드 주석 및 정리

5. README.md 작성 및 기능 설명

\*\*3️⃣ 정리할 학습 포인트\*\*

- GridView 성능 최적화 기법

- Provider 패턴의 상태 관리

- SharedPreferences 데이터 영속화

- 복잡한 비즈니스 로직 구현

- 이미지 처리와 캐싱 전략

\*\*4️⃣ 완성 기준\*\*

- 일관된 디자인과 사용자 경험

- 안정적인 에러 처리

- 완성된 README와 학습 정리

---

## \*\*🎯 학습 성과 평가\*\*

- \*\*기술적 완성도\*\*: 완전한 쇼핑몰 앱 기능 동작 🚀

- \*\*개념 적용도\*\*: 5개 복습 개념 완전 활용 💪

- \*\*코드 품질\*\*: 성능 최적화된 구조 📚

- \*\*학습 정리\*\*: 실무 패턴의 완전한 체득 ✍️

## \*\*💡 이번 프로젝트의 목적\*\*

- \*\*상태 관리 마스터\*\*: Provider 패턴을 활용한 복잡한 상태 관리 완전 체득

- \*\*실무 경험\*\*: 실제 전자상거래 앱 개발 프로세스 완전 체험

- \*\*성능 최적화\*\*: GridView, 이미지 처리 등 실무 필수 최적화 기법 습득

- \*\*비즈니스 로직\*\*: 장바구니, 주문, 계산 등 복잡한 비즈니스 로직 구현 능력 개발