Software Development

Cross-Platform Mobile App Development

DEPI 3rd Round 2025

Shop Smart App

Full System Documentation

Mobile Shopping Application

Supervised by:

Eng. Michael Fahmy

Team Members:

- Ahmed Nour
- Omar Mahmoud
- Osama Bhaa
- Mohamed Osama
- Mohamed Medhat
- Zakarya Hamdy

Table of Contents

This document outlines the key stages of the Shop Smart App project, from planning through system design.

4. UI/UX Design

- 4.1 Wireframes & User Flows
- 4.2 Screen Design
- 4.3 Color & Typography
- 4.4 Reusable Components

5. Application Structure

- 5.1 Project Structure
- 5.2 Application Setup
- 5.3 Data Flow & State Management
- 5.4 Screen Overviews
- 5.5 UI Components
- 5.6 Application Services

6. Firebase Integration

- 6.1 Authentication
- 6.2 Firestore Database
- 6.3 Real-time Data Sync
- 6.4 Security Rules

7. Testing & Quality Assurance

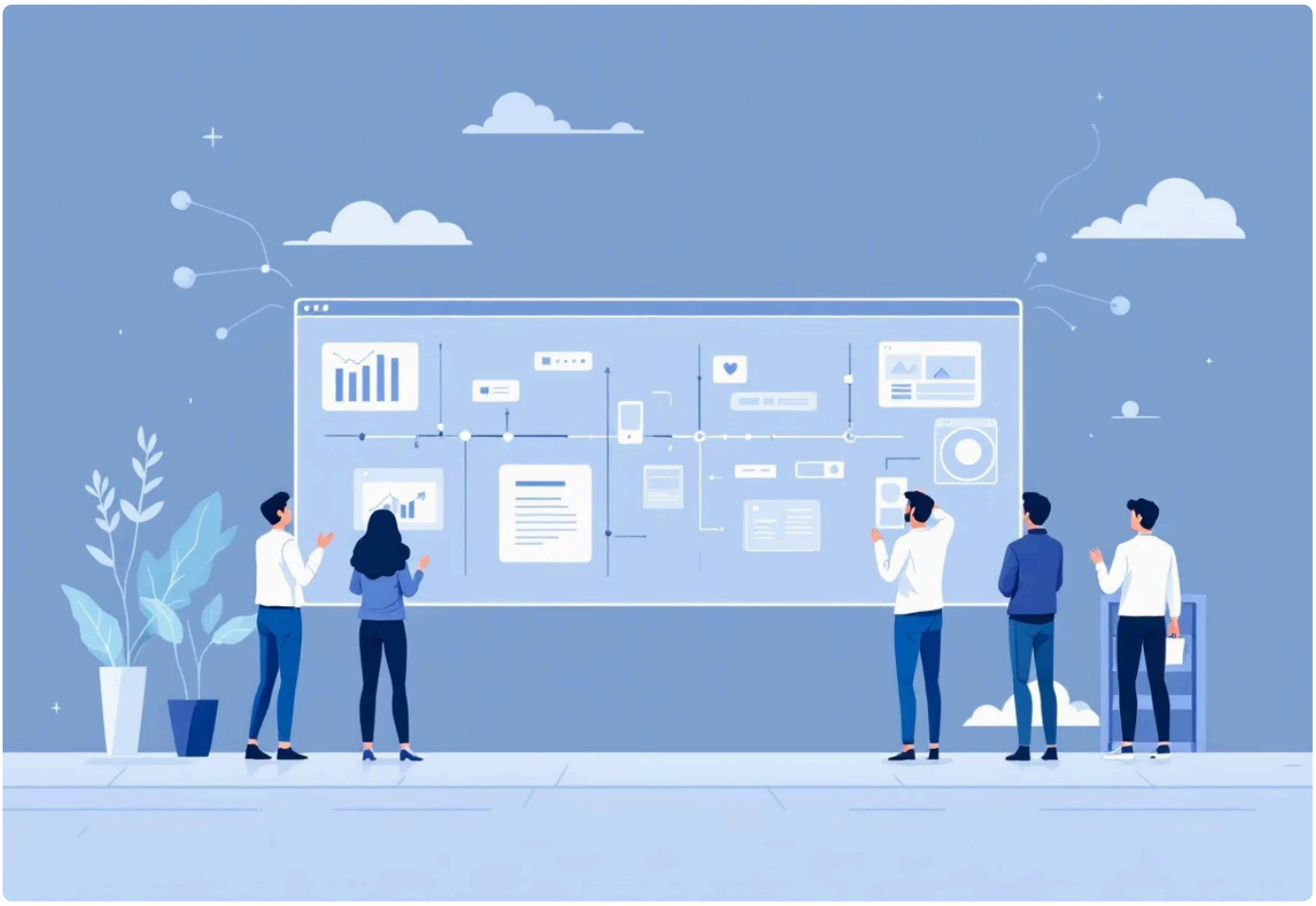
- 7.1 Test Planning
- 7.2 Test Cases
- 7.3 Bug Resolution

8. Deployment & Delivery

- 8.1 Technology Stack
- 8.2 Deployment Process
- 8.3 User Manual
- 8.4 Future Enhancements

9. Conclusion & Deliverables

1. Project Planning & Management



1.1 Project Proposal

The Shop Smart App is a comprehensive cross-platform mobile shopping solution designed to revolutionize the retail experience through robust backend integration and a seamless user experience.

Addressing mobile commerce demand, the app offers an intuitive platform for product browsing, cart management, payments, and user profiles. The project emphasizes clean architecture, scalable design, and modern development practices.

Project Vision: A user-centric mobile shopping application combining excellent UX with robust technical implementation, serving as a comprehensive learning platform for modern mobile development.

Project Mission: Develop a fully functional shopping app demonstrating proficiency in mobile app development, backend integration, data management, and UI/UX design principles.

1.2 Objectives

1

Primary Objectives

- Develop a complete mobile shopping application
- Implement robust backend integration (auth, data storage, real-time sync)
- Design scalable architecture for efficient data flow
- Design/implement responsive UI (Material Design)
- Establish secure user authentication and data management

2

Secondary Objectives

- Demonstrate modern mobile development proficiency
- Implement comprehensive testing
- Create detailed technical documentation
- Establish mobile application deployment procedures
- Develop reusable component libraries

3

Learning Objectives

- Gain expertise in mobile development frameworks/languages
- Understand BaaS concepts and integration
- Apply software engineering principles to mobile development
- Gain project management and team collaboration experience
- Develop UI/UX design and optimization skills

1.3 Scope of Work

In-Scope Features:

Core Functionality:

- Product browsing, search, and detailed views
- Shopping cart management
- User authentication and profile management
- Checkout & order tracking

Technical Implementation:

- Cross-platform mobile application
- Secure authentication & data storage
- Efficient state management
- Responsive UI with reusable components
- Form validation & error handling

Out-of-Scope Features:

- Real payment gateway integration
- Admin panel for product management
- Push notifications
- Social media integration
- Advanced analytics
- Multi-language support
- Offline functionality

1.4 Project Timeline

The project follows a 16-week development cycle, structured into four distinct phases:

01

Phase 1: Planning & Analysis (Weeks 1-4)

- Requirements, system design, and UI/UX planning.

02

Phase 2: Setup & Core Features (Weeks 5-8)

- Development environment, backend configuration, and basic architecture.

03

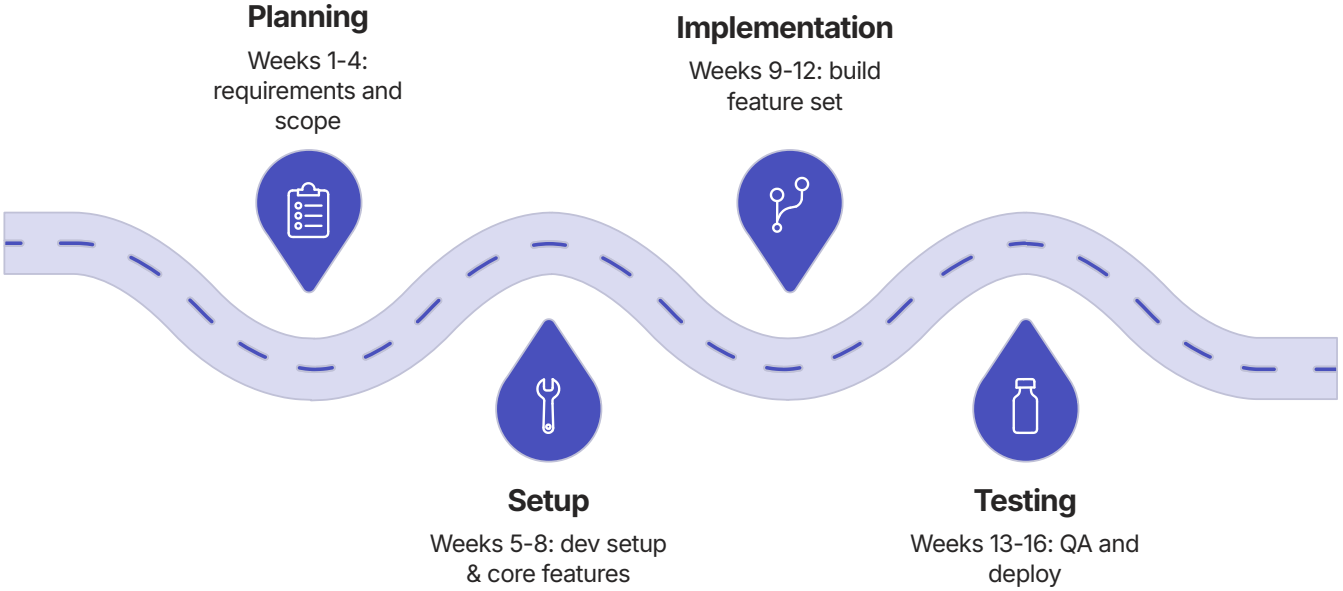
Phase 3: Feature Implementation (Weeks 9-12)

- Core shopping functionalities, including cart, profile, and checkout.

04

Phase 4: Testing & Deployment (Weeks 13-16)

- Quality assurance, issue resolution, and final deployment.



1. Project Planning & Management

1.5 Task Assignment

Ahmed Nour	**Team Leader & backend setup**
Omar Mahmoud	**Home / Auth Screens & presentation**
Osama Bhaa	**Profile screen & Reusable Components**
Mohamed Osama	**Documentation & Cart/Auth Screens**
Mohamed Medhat	**Theming & U**
Zakarya Hamdy	**Widgets & Testing**

1.6 Risk Assessment & Mitigation

Technical Risks:

High Priority

System Integration: Difficulties in core system setup.

- Mitigation: Early setup, documentation, local storage.

State Management: Challenges with chosen approach.

- Mitigation: Thorough study, incremental implementation, code reviews.

Cross-platform UI: Inconsistencies across devices.

- Mitigation: Multi-device testing, responsive design, best practices.

Medium Priority

Performance: Degradation with large datasets.

- Mitigation: Efficient loading, pagination, monitoring.

Third-party Dependencies: Breaking changes in updates.

- Mitigation: Version pinning, regular updates, fallback solutions.

Project Management Risks:

Team Coordination: Communication gaps, task overlap.

- Mitigation: Regular meetings, clear assignments, collaborative tools.

Timeline Delays: Feature implementation exceeding estimates.

- Mitigation: Buffer time, agile approach, priority-based development.

Scope Creep: Addition of unplanned features.

- Mitigation: Clear scope, change request process, stakeholder approval.

1.7 Key Performance Indicators (KPIs)

Development KPIs:

Code Quality & Stability:

- Robust, well-tested code
- Adherence to code review best practices
- High code maintainability
- Minimal defects
- Comprehensive internal documentation

Application Performance:

- App startup time: < 3s
- Screen transition: < 300ms
- Efficient resource utilization
- Minimal battery impact
- Optimized data communication

Development Process Efficiency:

- Sprint completion rate: 90%
- Feature delivery: 2-3 features/sprint
- Critical bug resolution: < 24h
- Consistent team contribution
- Automated core functionality testing

User Experience KPIs:

Usability Metrics:

- User task completion: 95% (core flows)
- Avg. purchase time: < 5 min
- Checkout error rate: < 2%
- Navigation efficiency: Max 3 taps/feature
- WCAG 2.1 AA compliance

User Satisfaction:

- App store rating target: 4.5+ stars
- User retention: 70% (after 1st week)
- Feature adoption (cart): 80%
- User feedback response: < 48h
- Crash rate: < 0.1%

2.2 User Stories

User stories capture functional requirements, grouped into epics, including acceptance criteria, priority, and effort.

Epic 1: User Authentication & Profile Management

As a new user, I want to create an account to access personalized shopping.

- Acceptance Criteria: Register with email/password, receive confirmation, access profile.
- Priority: High
- Estimated Effort: 8 story points

As a registered user, I want to log in to access my saved information.

- Acceptance Criteria: Log in with credentials, persist session, recover password.
- Priority: High
- Estimated Effort: 5 story points

As a user, I want to manage my profile to keep account details updated.

- Acceptance Criteria: View, edit, and save profile (name, email, preferences).
- Priority: Medium
- Estimated Effort: 6 story points

2.3 Use Cases & Detailed Scenarios

Use Case 1: User Registration and Authentication

Primary Actor: New User

Goal: Create an account and authenticate

Preconditions: App installed, internet connectivity

Success Scenario:

1. User opens app.
2. User selects "Create Account."
3. User provides email and password.
4. System verifies email format and password strength.
5. User confirms via email verification.
6. System creates profile and logs in user.
7. User directed to personalized home screen.

Alternative Scenarios:

- Email exists: system suggests login.
- Invalid email: validation error displayed.
- Weak password: user prompted to strengthen.
- Network error: retry option offered.

Postconditions: Account created; user authenticated.

Use Case 2: Product Search and Discovery

Primary Actor: Authenticated User

Goal: Find and view specific products

Preconditions: User logged in, internet connectivity

Success Scenario:

1. User accesses search function.
2. User enters search term.
3. System presents relevant results.
4. User explores results, using filters.
5. User selects a product for details.
6. System displays comprehensive product details.
7. User adds product to cart or continues browsing.

Alternative Scenarios:

- No results found: system suggests alternatives.
- Network timeout: system attempts cached results.
- Invalid search term: system provides suggestions.

Postconditions: User reviewed product information.

2.4 Functional Requirements

Functional requirements define what the system must do, outlining features necessary to satisfy user needs and business objectives.

Authentication & User Management

FR-001: User Registration

- Users can create accounts with email and password.
- Email format and password strength will be validated.
- Account activation requires email verification.
- Duplicate email registration will be prevented.

FR-002: User Authentication

- Users can authenticate with email and password.
- Persistent user sessions will be maintained.
- Password recovery functionality will be provided.
- Secure logout functionality will be implemented.

FR-003: Profile Management

- Users can view and modify profile information.
- Profile changes will be validated and securely stored.
- User preferences and settings will be displayed.
- Users can update their passwords.

Product Management

FR-004: Product Display

- Products will be displayed in a grid format.
- Product details (image, name, price, rating) will be shown.
- Categorization and filtering options will be available.
- Detailed product views with descriptions will be provided.

FR-005: Product Search

- A product search function will be available.
- Search results can be refined with filters and sorting.
- Search results will be ordered by relevance.
- Notification for no search results will be provided.

2.5 Non-Functional Requirements

Non-functional requirements define how the system performs, ensuring quality and user experience.

Performance Requirements

NFR-001: Response Time

- Home screen display within 3 seconds.
- Screen transitions within 300 milliseconds.
- Search results appear within 2 seconds.
- Shopping cart updates visible immediately.

NFR-002: Scalability

- Support up to 1000 simultaneous users.
- Manage product catalog of up to 10,000 items.
- Database queries maintain performance with data growth.
- Scalable via cloud-based infrastructure.

Usability Requirements

NFR-003: User Interface

- Adhere to design guidelines for consistency and usability.
- Adaptive interface for various screen sizes and orientations.
- Features accessible within 3 interactions.
- Clear visual cues for user actions.

NFR-004: Accessibility

- Integrate with screen readers and assistive technologies.
- Text meets minimum contrast for readability.
- Interactive elements sized for easy touch.
- Adjustable text sizing based on user preference.

3. System Analysis & Design

3.1 Problem Statement

Existing e-commerce platforms struggle to deliver seamless, high-performance mobile shopping experiences due to complex interfaces and fragmented designs.

Core Problems Identified:

User Experience Challenges:

- Complex navigation and inconsistent design
- Poor performance causing user frustration
- Lack of real-time feedback
- Inadequate mobile optimization

Technical Implementation Issues:

- Monolithic architecture hinders scalability
- Poor state management
- Inadequate separation of concerns
- Limited UI component reusability
- Insufficient error handling

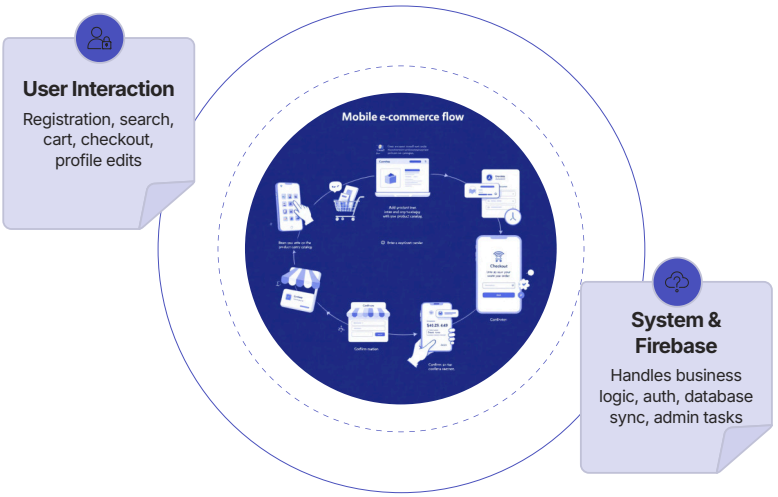
Business Requirements:

- Need for rapid development/deployment
- Cross-platform compatibility
- Scalable backend infrastructure
- Modern authentication/data management integration
- Real-time sync and offline capabilities

Proposed Solution:

The Shop Smart App addresses these challenges with a modern Flutter-based mobile application, clean architecture, Firebase for backend, robust state management, and Material Design principles, ensuring performance and user satisfaction.

3.2 Use Case Diagram



This use case diagram outlines primary interactions between users, the Shop Smart App system, and Firebase backend services, detailing key functional areas and relationships.

3.3 System Architecture

The Shop Smart App employs a layered architecture, leveraging Flutter's reactive model and Firebase integration, to ensure separation of concerns, maintainability, and scalability.

Architecture Layers:

Presentation Layer (UI)

- User interface, visual components
- Theme, input handling
- Navigation

Business Logic Layer (State Management)

- Application state management
- Business rules, data processing
- Event coordination

Data Access Layer (Services)

- Firebase and API integration
- Data caching and synchronization
- Error handling, security services

Backend Infrastructure (Firebase)

- User authentication
- NoSQL database, media storage
- Server-side logic, performance tracking

3.4 Database Design

The Shop Smart App utilizes Firebase Firestore for its primary database, leveraging real-time synchronization, offline support, and scalable NoSQL document storage. The design follows a denormalized approach optimized for mobile application performance.

Firestore Collections Structure:

Users Collection

Stores user profiles including email, display name, profile image URL, preferences, multiple addresses, and timestamps.

Products Collection

Contains product details such as name, description, price, category, image URLs, ratings, stock information, and timestamps.

Cart Items Subcollection

Nested within user profiles, this subcollection manages items in a user's cart, including product ID, name, price, image URL, quantity, and timestamps.

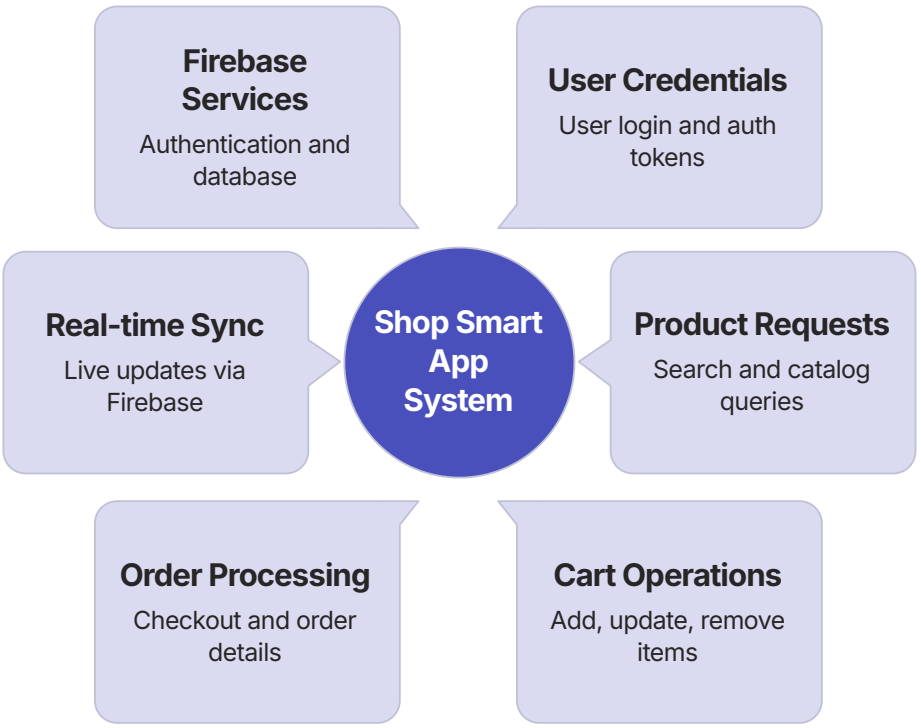
Orders Collection

Records all customer orders, detailing the user, order number, status, total amount, purchased items (product ID, name, quantity, price), shipping address, payment method, and timestamps.

3.5 Data Flow Diagrams

Level 0 Data Flow Diagram (Context Diagram)

The Level 0 DFD illustrates the Shop Smart App system's high-level interactions with external entities.



External Entities:

- **User**: For all shopping activities.
- **Firebase Services**: Backend for authentication, data, and real-time sync.

Key Data Flows:

- User authentication and profile.
- Product browsing and search queries.
- Shopping cart operations.
- Order placement and payment.
- Real-time data synchronization.

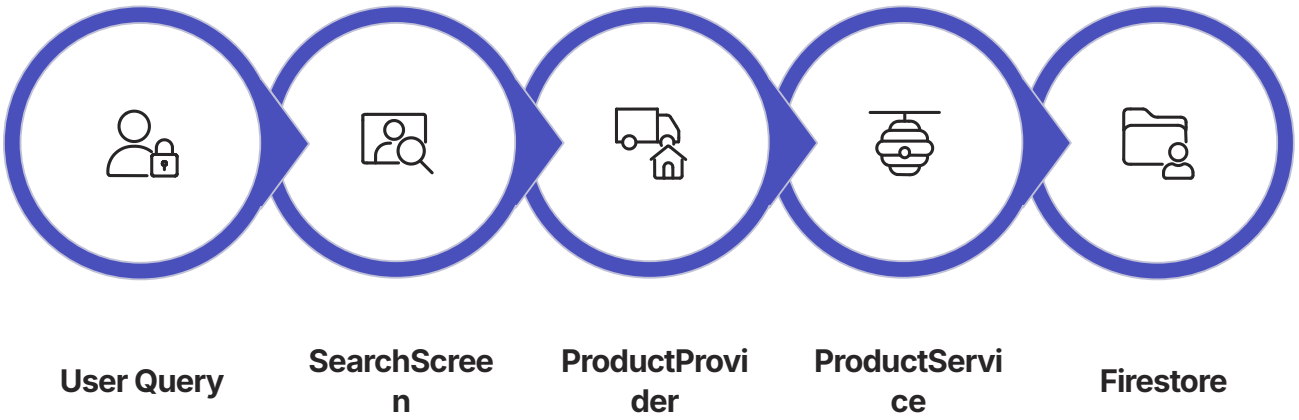
3.6 Sequence Diagrams

User Authentication Sequence

Authentication Flow:

1. User submits credentials via LoginScreen.
2. LoginScreen initiates authentication via AuthProvider.
3. AuthProvider validates credentials using FirebaseAuth.
4. FirebaseAuth authenticates user, returning a token.
5. AuthProvider updates application state; UserService loads profile.
6. UI navigates to home screen.

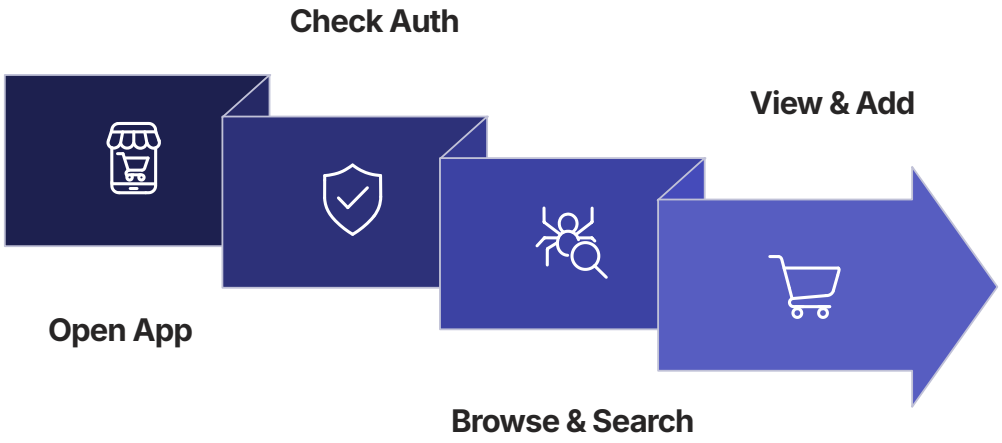
Product Search and Selection Sequence



3.7 Activity & State Diagrams

Product Browsing Activity Diagram

This diagram outlines the core user flow for product browsing in the Shop Smart App, from launch to cart addition or continued browsing.

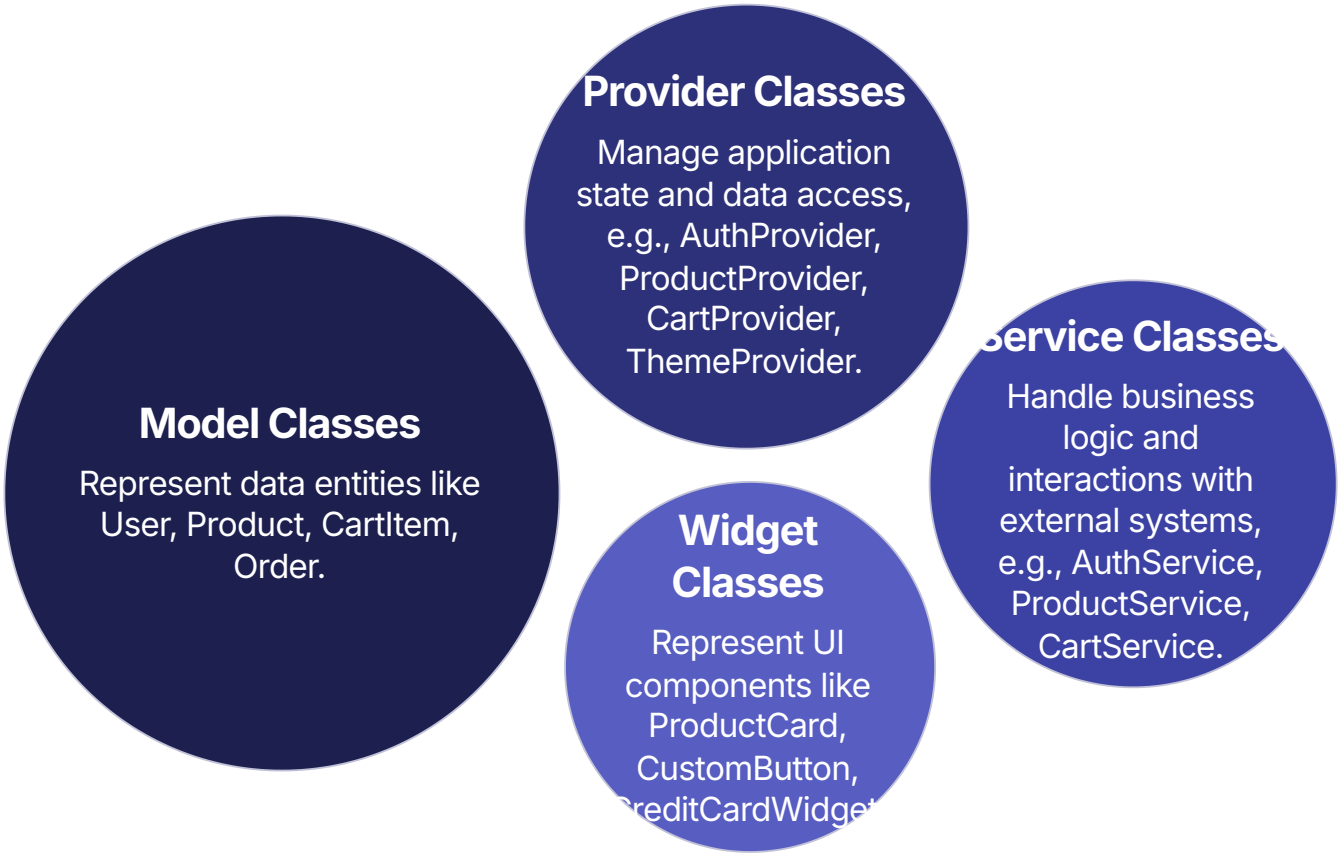


Product Browsing Activities:

1. App Launch & User Authentication
2. Display Home Screen/Categories
3. Browse/Search Products
4. View Product Details
5. Add Product to Cart (Optional)
6. Confirmation & Continue/Exit

3.8 Class Diagram

This class diagram outlines the object-oriented architecture of the Shop Smart App, detailing core classes, their relationships, attributes, and methods.



4. UI/UX Design

4.1 Wireframes & User Flow

The Shop Smart App's UI adheres to Material Design principles for consistency and usability. Wireframes define the application's layout, navigation, and content hierarchy.

Key User Flows:

Onboarding:

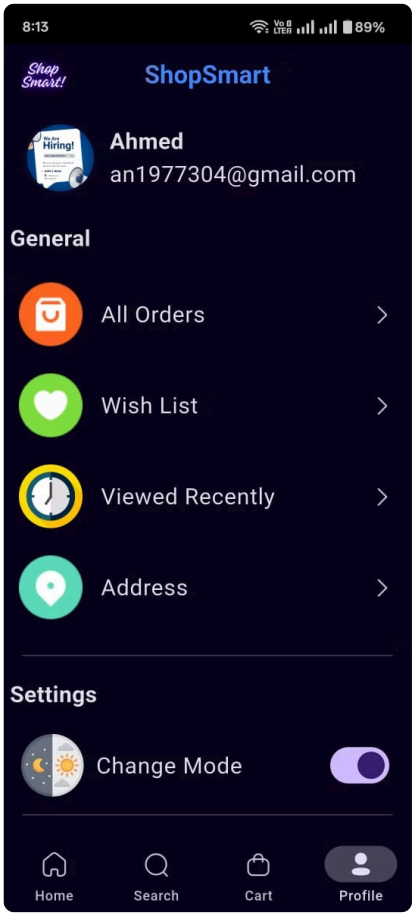
1. Splash → Welcome → Login/Register → Home
2. Guest Access → Limited Home → Registration Prompt

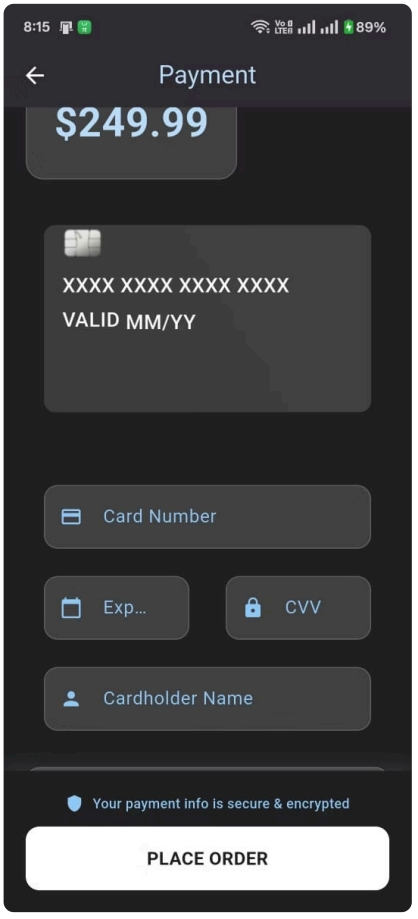
Shopping:

1. Home → Category/Search → Product List → Product Details → Add to Cart
2. Cart Review → Checkout → Payment → Order Confirmation

Profile Management:

1. Home → Profile → Edit Profile → Save Changes → Confirmation





4.2 Screen Design Analysis

Home Screen Design:

Layout Structure:

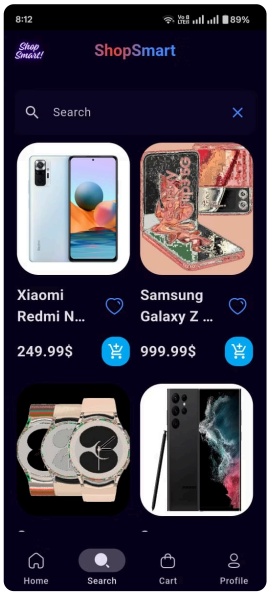
- App Bar (search icon, cart badge)
- Featured products carousel
- Category grid (visual icons)
- Product grid (infinite scroll)
- Bottom navigation bar

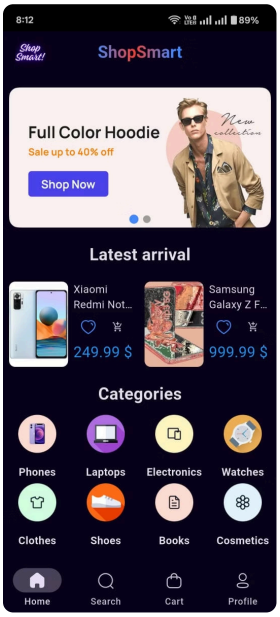
Key Components:

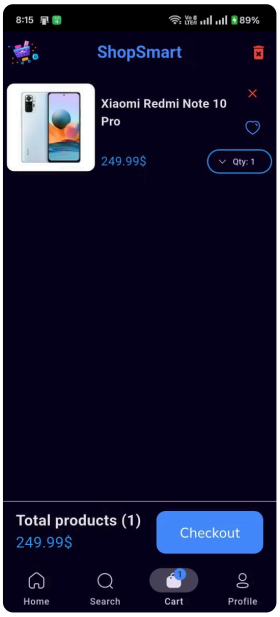
- Product cards (image, name, price, rating)
- Category tiles (images)
- Search bar (voice input)
- Cart icon (item count)

User Interactions:

- Product card: Navigates to product details
- Category: Filters products
- Search: Opens search interface
- Cart: Navigates to shopping cart







4. UI/UX Design

4.3 Color Theme & Typography

Color Palette Design:

The Shop Smart App utilizes a balanced color palette for optimal readability and visual hierarchy.

Primary Colors

- **Primary Blue**: #2196F3 - For primary actions, links, and branding.
- **Primary Dark**: #1976D2 - For app bars, status bars, and emphasis.
- **Primary Light**: #BBDEFB - For backgrounds and subtle highlights.

Secondary Colors

- **Accent Orange**: #FF9800 - For call-to-action buttons and notifications.
- **Accent Dark**: #F57C00 - For hover states and active elements.
- **Accent Light**: #FFE0B2 - For subtle backgrounds and highlights.

Neutral Colors

- **Text Primary**: #212121 - For headings and important content.
- **Text Secondary**: #757575 - For descriptions and metadata.
- **Divider**: #BDBDBD - For separators and borders.
- **Background**: #FAFAFA - Primary background color.
- **Surface**: #FFFFFF - Card and surface backgrounds.

Semantic Colors

- **Success Green**: #4CAF50 - For success messages.
- **Warning Amber**: #FFC107 - For warning messages.
- **Error Red**: #F4436 - For error messages and destructive actions.
- **Info Blue**: #2196F3 - For informational messages.

4.4 Reusable Components

The Shop Smart App utilizes a comprehensive component library for consistency, maintainability, and development efficiency.

Core UI Components:

ProductCardWidget:

- Displays product info (image, name, price, rating)
- Supports various sizes
- Handles loading/error states and tap gestures

CustomTextWidget:

- Standardized text component with design system styling
- Handles overflow, accessibility, and theme-aware colors

CreditCardWidget:

- Interactive visualization with card flip animation
- Real-time card number validation
- Displays card brand logos

CustomSnackBarWidget:

- Consistent notification system for various message types (success, error, info)
- Supports auto-dismissal and user actions

CustomEmptyWidget:

- Standardized empty state display for various scenarios
- Includes illustrations, messaging, and optional CTAs

5. Implementation Details

5.1 Project Structure Overview

The Shop Smart App employs a modular project structure, designed for maintainability, scalability, and collaborative development. This architecture effectively separates concerns, ensuring a clear and navigable codebase.

Modular Project Structure:

The project is organized into distinct modules: state management providers, feature-grouped UI screens, reusable UI components (widgets), business logic and external integrations (services), application constants, and data models. The main application entry point and routing tie these modules into a coherent structure.

Architecture Benefits:

Separation of Concerns:

- UI components isolated (screens/, widgets/)
- Business logic centralized (providers/, services/)
- Data models defined separately (models/)
- Constants and configuration organized (consts/)

Scalability:

- Feature-based organization for easy module addition
- Modular structure supports team development
- Clear layer dependencies
- Easy modification of specific functionality

Maintainability:

- Consistent naming conventions
- Logical grouping of related functionality
- Clear separation of presentation and business logic
- Standardized patterns

5.2 Main Application Setup

The `main.dart` file initializes the app, configuring providers, themes, and routing.

Key Initialization Components:

1

Firebase Initialization:

- Initializes Firebase services, including authentication and Firestore.

2

Provider Setup:

- MultiProvider manages app-level state for global access to domain logic.

3

Theme Configuration:

- Sets up comprehensive light and dark themes with consistent typography.

4

Routing Configuration:

- Centralized route management via AppRoutes, supporting named routes and parameter passing.

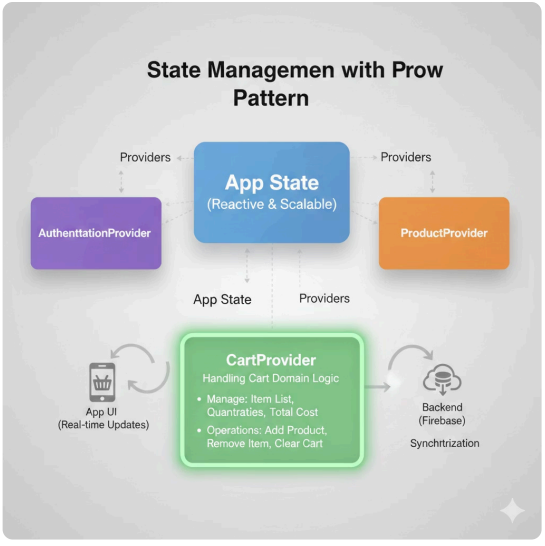
5.3 State Management with Providers

The Shop Smart App utilizes the Provider pattern for reactive, scalable, and testable state management, ensuring clear separation of concerns.

CartProvider Functionality:

Manages all shopping cart operations, including:

- Adding, updating, and removing items
- Calculating total costs
- Synchronizing cart state with Firebase for real-time UI updates.



6. Firebase Integration

Authentication:

- Email/password authentication
- User session management
- Secure token handling

Firestore Database:

- Real-time product data synchronization
- User profile and cart storage
- Offline data caching

Storage:

- Product image hosting
- User profile pictures
- Optimized image delivery

Security Rules:

- User-specific data access
- Read/write permissions
- Data validation

7. Testing Strategy

The Shop Smart App employs a robust testing strategy to ensure high quality, reliability, and performance across all functionalities and user interactions.

Unit Testing:

- Provider logic testing
- Service layer validation
- Widget functionality tests

Integration Testing:

- Firebase connectivity
- Authentication flow
- Data synchronization

User Acceptance Testing:

- Core user journeys
- Performance benchmarks
- Cross-device compatibility

Test Coverage Goals:

- 80% code coverage minimum
- All critical paths tested
- Automated testing pipeline

8. Deployment & Distribution

Development Environment:

- Flutter SDK setup
- Firebase project configuration
- Development device testing

Production Build:

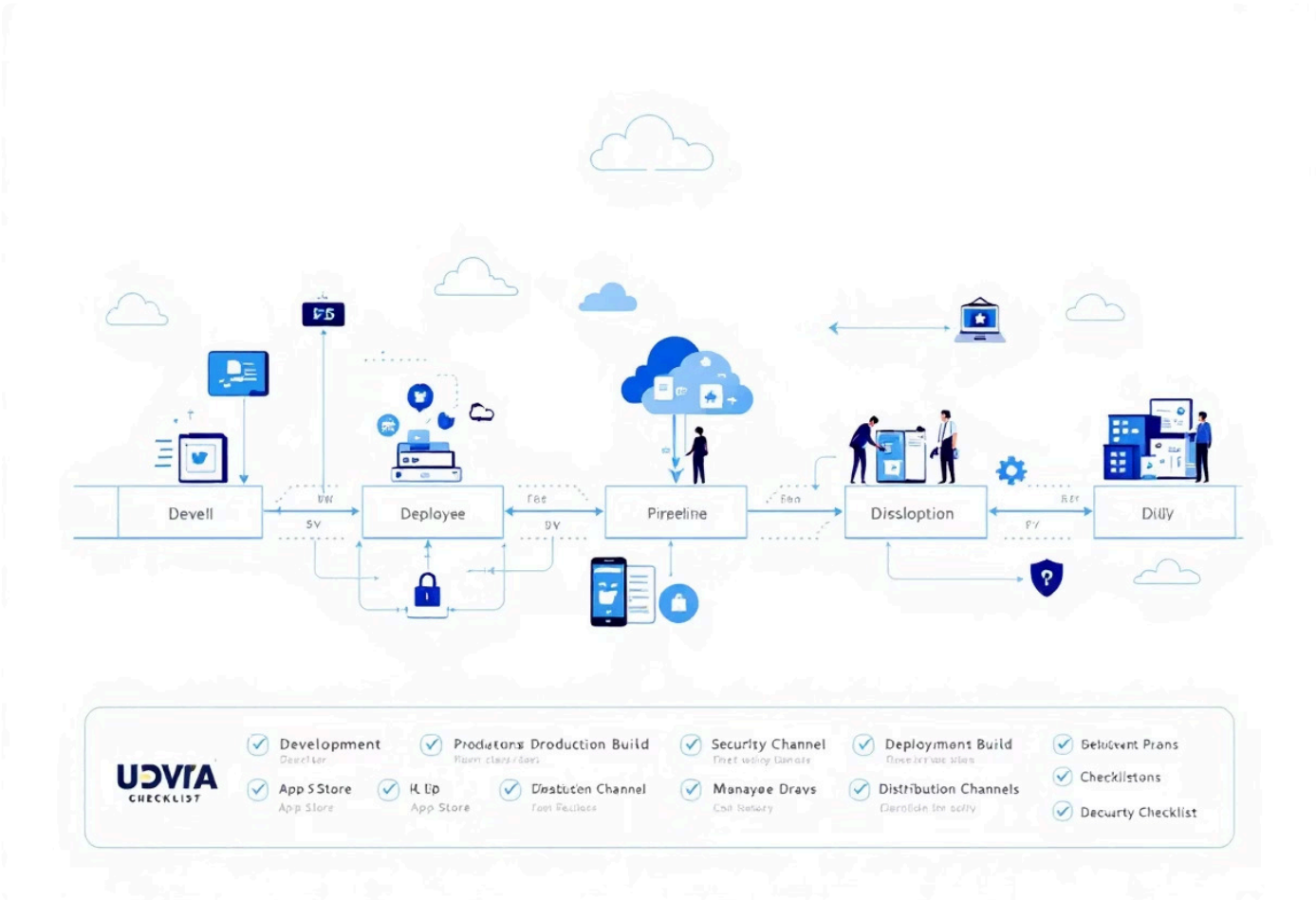
- Release APK generation
- Code obfuscation
- Performance optimization

Distribution Channels:

- Google Play Store
- Internal testing via Firebase App Distribution
- Beta testing program

Deployment Checklist:

- Security audit completion
- Performance testing validation
- Store listing optimization
- User documentation



9. Project Summary & Conclusion

Project Achievements:

- Complete cross-platform mobile shopping application
- Robust Flutter + Firebase architecture
- Comprehensive user authentication and data management
- Professional UI/UX following Material Design principles

Technical Deliverables:

- Fully functional mobile application
- Scalable backend infrastructure
- Comprehensive documentation
- Testing framework implementation

Learning Outcomes:

- Mobile app development expertise
- Cloud integration proficiency
- State management mastery
- Professional development practices

Future Enhancements:

- Payment gateway integration
- Advanced recommendation system
- Multi-language support
- Analytics dashboard