

# Penny - AI-First Budgeting App

## Project Document for Kiro Development

---

### Executive Summary

**Penny** is a revolutionary iOS budgeting app that eliminates traditional budgeting friction through AI automation and camera-based affordability checking. Built for the 2026 Swift Student Challenge, Penny represents the future of personal finance management where AI assistance meets absolute privacy through on-device processing.

**Core Mission:** Transform budgeting from a tedious chore into an effortless, intelligent experience using Apple's latest Foundation Models and VisionKit frameworks.

---

### Unique Value Proposition

#### "Point, Ask, Budget Smart"

##### Four Revolutionary Features:

1. **Instant Affordability** - Point camera at any item → immediate yes/no purchase decision
  2. **Zero Budget Management** - AI automatically adjusts spending limits based on your behavior
  3. **Effortless Tracking** - Apple Pay-style input makes expense entry seamless
  4. **Privacy-First Intelligence** - All AI processing happens on your device, never in the cloud
- 

### The Problem We're Solving

#### Current budgeting apps fail because they:



- Require manual budget setup and maintenance
- Make expense tracking tedious and time-consuming
- Provide generic advice that doesn't adapt to individual behavior
- Compromise user privacy by processing financial data in the cloud
- Create decision fatigue with complex categories and rules

**Penny's Solution:** An intelligent financial assistant that learns your spending patterns, automatically manages your budget, and provides instant purchasing guidance through your camera - all while keeping your data completely private on your device.

---

# Design Philosophy

## Visual Identity

- **Style:** Neo-brutalist minimalism with premium feel
- **Colors:** Electric Blue ( #007AFF), Success Green ( #34C759), Pure White backgrounds
- **Typography:** SF Pro Display (headers), SF Mono (financial numbers)
- **Interactions:** Smooth spring animations, haptic feedback, micro-interactions

## UX Principles

- **Information Hierarchy:** Balance > Affordability > Categories > History
  - **One-Handed Usage:** Critical actions within thumb reach
  - **Cognitive Load:** Maximum 3 primary actions per screen
  - **Progressive Disclosure:** Advanced features hidden until needed
- 

## Core Features

### 1. Camera Affordability Scanner **UNIQUE FEATURE**

- Full-screen VisionKit camera interface
- Real-time object recognition and price estimation
- Instant affordability calculation against current budget
- Slide-up result card with color-coded feedback
- Quick "Add to Budget" action integration

#### User Flow:

1. Open app → Tap camera button
2. Point camera at item you want to buy
3. Get instant "CAN AFFORD" or "SKIP THIS" recommendation
4. Optional: Add purchase to budget with one tap

### 2. AI Budget Auto-Management **CORE INTELLIGENCE**

- Foundation Models analyze spending patterns
- Automatic budget redistribution between categories
- Smart notifications for unusual spending

- Learning algorithms that improve over time
- Natural language expense processing

### Example AI Insights:

- "You typically spend 20% more on groceries in winter months"
- "Moving \$50 from entertainment to groceries based on your recent patterns"
- "Great job! You're on track to save an extra \$200 this month"

### 3. Apple Pay-Style Input 📱 PREMIUM UX

- Exact replica of Apple Pay's number pad interface
- Large amount display with elegant typography
- Circular number buttons with haptic feedback
- Category selection with smooth animations
- Modal presentation with blur effects

### 4. Duolingo Streak System 🔥 GAMIFICATION

- Circular progress ring with animated fire emoji
- Track consecutive days within AI-adjusted budget
- Milestone celebrations with confetti effects
- Achievement badges and streak protection
- Weekly calendar visualization

### 5. Face ID Private Mode 🔒 PRIVACY & SECURITY

- Toggle to obfuscate all financial data with "...."
- Face ID/Touch ID authentication to disable private mode
- Smooth blur/unblur transition animations
- Secure data handling with Secure Enclave integration

---

## 🏗️ Technical Architecture

### Core Stack

- **Platform:** iOS 17+ with SwiftUI
- **Architecture:** MVVM with ObservableObject

- **Persistence:** AppStorage with CryptoKit encryption
- **AI:** Apple Foundation Models (on-device)
- **Vision:** VisionKit for camera features
- **Auth:** LocalAuthentication (Face ID/Touch ID)

## Key Data Models

swift

```
struct Transaction: Identifiable, Codable {  
    let id = UUID()  
    let amount: Double  
    let category: Category  
    let isIncome: Bool  
    let date: Date  
    let note: String?  
    let aiConfidence: Double  
}
```

```
struct AIBudget: Codable {  
    var totalMonthlyBudget: Double  
    var categoryLimits: [Category: Double]  
    var aiAdjustments: [BudgetAdjustment]  
    var lastOptimization: Date  
}
```

```
struct StreakData: Codable {  
    var currentStreak: Int  
    var longestStreak: Int  
    var streakHistory: [Date: Bool]  
    var milestones: [StreakMilestone]  
}
```

## AI Integration Patterns

swift

```
// Foundation Models Usage
import FoundationModels

class AIBudgetManager: ObservableObject {
    func optimizeBudget() async -> [BudgetAdjustment]
    func processNaturalLanguage(_ text: String) -> Transaction?
    func generateInsights() -> [FinancialInsight]
    func predictAffordability(_ item: RecognizedItem) -> Bool
}

// VisionKit Implementation
import VisionKit

class AffordabilityScanner: ObservableObject {
    func scanForAffordability(_ image: UIImage) async -> AffordabilityResult
    func recognizePrice(in image: UIImage) -> Double?
    func classifyObject(in image: UIImage) -> String?
}
```

## App Structure

### Main Navigation (TabView)

1. **Dashboard** - Balance, camera button, quick insights
2. **Budget** - AI management, category progress, settings
3. **History** - Transaction list, search, analytics
4. **Settings** - Privacy, notifications, preferences

### Primary Views

- **ContentView** - Tab container
- **DashboardView** - Main balance and camera access
- **AffordabilityView** - Full-screen camera scanner
- **TransactionInputView** - Apple Pay-style input
- **BudgetIntelligenceView** - AI insights and controls
- **StreakView** - Gamification dashboard
- **PrivacyView** - Face ID and data security

# Development Roadmap

## Phase 1: Core Foundation **COMPLETED**

- Basic SwiftUI app structure and navigation
- Core transaction tracking with categories
- Monthly budget tracking with visual progress
- Duolingo-style streak counter implementation
- Data persistence with AppStorage

## Phase 2: Intelligence Features **IN PROGRESS**

- VisionKit Camera Integration
- Apple Pay Input Interface
- Foundation Models AI Integration
- Face ID Private Mode
- Advanced streak gamification

## Phase 3: Polish & Optimization

- Cross-platform responsive layouts (iPad, Mac, Apple Watch)
  - Comprehensive accessibility support
  - Performance optimization and testing
  - Advanced AI features and insights
- 

## Success Metrics

### Innovation Showcase

- First budgeting app using Apple Foundation Models
- Unique camera-based affordability checking
- Advanced on-device AI implementation
- Privacy-first architecture demonstration

### Technical Excellence

- Clean, maintainable SwiftUI code
- Proper use of latest Apple frameworks

- Outstanding performance optimization (<2s launch, <1s camera response)
- Comprehensive accessibility support

## Real-World Impact

- Genuinely useful financial management tool
  - Intuitive user experience design
  - Practical AI assistance that improves over time
  - Measurable behavior change potential
- 

## Privacy & Security

### Privacy-First Architecture:

- All AI processing happens on-device using Apple's Foundation Models
- Financial data never leaves the user's device
- CryptoKit encryption for all stored data
- Face ID/Touch ID for sensitive operations
- No user accounts or cloud sync required

**User Promise:** "Your money, your device, your privacy"

---

## Business Model & Market Opportunity

### Target Market

- Tech-savvy millennials and Gen Z users
- iOS users who value privacy and premium experiences
- People frustrated with traditional budgeting apps
- Early adopters of AI-powered financial tools

### Monetization Strategy

- Premium subscription for advanced AI features
  - One-time purchase for full feature unlock
  - Potential partnership with financial institutions
  - Focus on user value over aggressive monetization
-

## Brand Voice & Messaging

**Tone:** Confident, helpful, privacy-focused, innovative

### Key Messages:

- "Your smartest financial decision starts with Penny"
  - "Budgeting made effortless through intelligent automation"
  - "Point your camera, make smart choices"
  - "AI that works for you, not against your privacy"
- 

## Implementation Guidelines

### Code Quality Standards

- SwiftUI best practices with proper state management
- MVVM separation with business logic in ViewModels
- Full VoiceOver support throughout the app
- Comprehensive error handling and edge cases
- Performance targets: <2s launch, <1s camera response, <500ms AI processing

### Design Standards

- Follow Apple's Human Interface Guidelines
  - Implement smooth spring animations and haptic feedback
  - Design for one-handed mobile usage
  - Maintain consistency across all platforms
  - Prioritize accessibility in every design decision
- 

## Why Penny Will Win

**Technical Innovation:** First to combine Apple's Foundation Models with VisionKit for real-time affordability checking

**User Experience:** Eliminates the friction that makes people abandon budgeting apps

**Privacy Leadership:** Sets new standard for financial app privacy with on-device AI

**Market Timing:** Perfect alignment with Apple's AI capabilities and growing privacy concerns



**Real Utility:** Solves actual problems instead of adding complexity

---

*This app represents the future of personal finance - where AI assistance meets absolute privacy, and budgeting becomes effortless rather than overwhelming.*