

Finance AI App Core Guidelines for each part:


Make sure to pull and push every time when working on the project:


1. Budgeting & Expense Tracking Section

Goal:

Help users understand, track, and forecast their spending habits — even if they don't manually log every purchase.

✅ Features to Implement:

- **Manual Expense Input**
 - Fields: amount, category, notes, date
 - Category options: Food, Travel, Shopping, Bills, Subscriptions, Other
 - Store data locally using **Core Data**
- **Smart Input Options**
 -  **Bank Account Integration** (*optional*) via Plaid or Apple's Financial Data APIs

Let users sync transaction history securely
 -  **Receipt Scanning with OCR**

Use iPhone camera + VisionKit or Firebase ML to extract total, date, category
 - **Apple Pay Notifications Parsing**

Auto-log transactions from Apple Wallet notifications (with user permission)

- **Recurring Expenses**
 - Toggle for “repeat monthly”
 - Stored in a separate model and auto-logged on schedule
- **Auto-Categorization (NLP-powered)**
 - Use keywords or merchant name to classify transactions
 - Example: “Spotify” → Subscription, “Chipotle” → Food
- **Monthly Budget Setup**
 - Set limits per category
 - Visual feedback with bar charts and indicators (via Swift Charts)
- **Spending Insights**
 - Show weekly/monthly summaries, “biggest spend,” category breakdowns
 - Forecast future expenses using trend-based prediction (linear regression or simple ML)

Core Tools:

- SwiftUI (UI)
- Core Data (storage)
- VisionKit or Firebase ML (receipt OCR)
- Apple Notification Service Extension (Apple Pay parsing)
- Optional: Plaid API (bank connection)

How to Code It:

Manual Entry Form

swift

CopyEdit

- `@State private var amount: String = ""`
- `@State private var category: String = "Food"`
- `@State private var note: String = ""`
-
- `TextField("Amount", text: $amount)`
- `Picker("Category", selection: $category) {`
- `ForEach(categories, id: \.self) { Text($0) }`
- `}`
- `TextField("Note", text: $note)`
- `Button("Add") {`
- `let newExpense = Expense(context: viewContext)`
- `newExpense.amount = Double(amount) ?? 0.0`
- `newExpense.category = category`
- `newExpense.note = note`
- `try? viewContext.save()`
- `}`

Apple Pay Notification Parser (Optional)

- Use **Notification Service Extension**
- Parse notification content with regex
- Example:

swift

CopyEdit

- `if notification.body.contains("Apple Pay") {`
- `// Extract amount + merchant`
- `}`

Receipt Scanner

swift

CopyEdit

- `import VisionKit`
-
- `let scannerVC = VNDocumentCameraViewController()`
- `scannerVC.delegate = self`
-
- `// Then use Vision/TextRecognition to parse totals and dates`

✓ **Bank Connection (Optional)**

- Integrate **Plaid SDK**
 - Fetch transactions → classify → store in Core Data
-

2. Stocks Section

Goal:

Allow users to track stocks, visualize trends, and optionally receive predictions or alerts — all without needing to invest.

✓ **Features to Implement:**

- **Stock Search + Watchlist**
 - Use **Yahoo Finance API** or **Alpha Vantage**
 - Let users save tickers to a watchlist (AAPL, TSLA, etc.)
- **Real-Time Price Updates**
 - Show price, % change, volume, market cap
 - Line chart of price history (1D, 1W, 1M)
- **Beginner-Friendly Summaries**
 - “AAPL is a tech company. Its stock has gone up 15% this month.”

- Integrate with Finbot or show via pre-written summaries
- **Simple Prediction Tool (Optional)**
 - Run simple LSTM/ARIMA model (can start offline)
 - Show risk/confidence meter: “Moderate confidence in a price increase”

Core Tools:

- SwiftUI, Swift Charts
- Yahoo Finance API / Alpha Vantage
- Alamofire or URLSession

How to Code It:

Fetch Stock Data

swift

CopyEdit

- `import Alamofire`
-
- `func fetchStockData(for ticker: String) {`
- `let url =`
`"https://query1.finance.yahoo.com/v7/finance/quote?symbols=\(tick`
`er)"`
- `AF.request(url).responseJSON { response in`
- `// Parse and update UI`
- `}`
- `}`

Chart Display

swift

CopyEdit

- `import Charts`
-
- `LineChart(data: stockPrices)`
- `.frame(height: 200)`

✓ Watchlist Logic

- Use `@AppStorage` or Core Data to save user-selected tickers
 - Render in a `List`
-

3. Finbot – AI Financial Assistant

Goal:

Offer a conversational interface for users to ask financial questions, track their money, and get friendly reminders.

✓ Features to Implement:


- **Chat Interface (like a messaging app)**
 - Allow user input + responses via OpenAI API or Firebase NLP
 - Suggested prompts:
 - “What did I spend the most on last week?”
 - “How much do I have left in my food budget?”
 - “What does ‘dividend’ mean?”
- **Smart Alerts**
 - If budget exceeds, Finbot sends:

“You’ve exceeded your food budget by \$23 this week.”

- Upcoming bill reminders
- **Tone & Personality**
 - Friendly, motivating (e.g., “Nice savings this week!”)
 - Preload fallback responses for offline use

How to Code It:

-  **Basic Chat Interface**
- swift
- CopyEdit
- `@State var messages: [String] = []`
- `@State var input: String = ""`
-
- `TextField("Ask Finbot...", text: $input)`
- `Button("Send") {`
- `messages.append(input)`
- `callOpenAI(input)`
- `}`
-
-  **Call OpenAI API**
- swift
- CopyEdit
- `func callOpenAI(_ prompt: String) {`
- `let headers = [`
- `"Authorization": "Bearer YOUR_API_KEY",`
- `"Content-Type": "application/json"`
- `]`
- `let body = [`
- `"model": "gpt-4",`
- `"messages": [{"role": "user", "content": prompt}]`
- `]`
-
- `AF.request("https://api.openai.com/v1/chat/completions",`
- `method: .post,`
- `parameters: body,`
- `encoding: JSONEncoding.default,`
- `headers: HTTPHeaders(headers)).responseJSON {`
- `response in`

- `// Parse and show Finbot response`
 - `}`
 - `}`
 -  **Suggested Prompts**
 - swift
 - CopyEdit
 - `let quickPrompts = ["What did I spend most on?", "How much is left in my budget?"]`
-


4. Financial Quest – Learn & Level Up

Goal:

Gamify finance learning with daily missions, short lessons, and progress badges.

Features to Implement:

- **Mini Lessons**
 - Topics: Budgeting basics, What is a stock?, How to save smarter
 - Format: 3–5 slides per lesson with images + quiz
- **Quizzes & Challenges**
 - Example: “Guess which subscription you’ve spent most on?”
 - Score XP and unlock “badges”
- **Progress Tracker**
 - Show streaks, level-up bar, completed missions

 Optional: Let Finbot recommend quests based on spending patterns

Core Tools:

- SwiftUI pages

- Local JSON for quiz content
- GameKit or in-app data for XP/badges

How to Code It:

✓ Slide-Based Lessons

swift

CopyEdit

```
TabView {  
    ForEach(lessons) { slide in  
        VStack {  
            Text(slide.title)  
            Image(slide.imageName)  
            Text(slide.description)  
        }  
    }  
}  
  
.tabViewStyle(PageTabViewStyle())
```

✓ XP Tracking

- Create a simple `UserProgress` model
- Store lesson completion using `@AppStorage` or Core Data


5. 🖐️ Intro App Tour (Onboarding New Users)

Goal:

Quickly show users what the app can do and help them set up their dashboard.

Features to Implement:

- **Welcome Carousel**
 - 3–5 screens explaining:
 - Expense Tracking
 - Finbot AI
 - Stocks Dashboard
 - Smart Alerts & Privacy Options
- **Quick Setup**
 - Optionally ask:
 - Monthly income range
 - Financial goals (save, track, invest)
 - Notification preferences
 - Skip option for minimal friction
- **Privacy/Trust Message**
 - “You control your data. Bank connection optional.”
 - Earns user trust early

 Optional: Show a friendly greeting from Finbot at the end of onboarding

Core Tools:

- SwiftUI `TabView`

- `AppStorage` for “hasSeenOnboarding” flag

How to Code It:

swift

CopyEdit

```
@AppStorage("hasSeenOnboarding") var hasSeenOnboarding: Bool = false
```

```
if !hasSeenOnboarding {  
    TabView {  
        OnboardingPage(title: "Track Expenses", image: "money")  
        OnboardingPage(title: "Meet Finbot", image: "chatbot")  
        OnboardingPage(title: "Watch Stocks", image: "stocks")  
    }  
    .tabViewStyle(PageTabViewStyle())  
    Button("Get Started") {  
        hasSeenOnboarding = true  
    }  
} else {  
    MainDashboard()  
}
```

Tools + Tech Checklist

Feature	Tools / APIs Used
UI	SwiftUI, Figma (design), Xcode
Data Storage	Core Data, Firebase (optional)
API Integration	Yahoo Finance API, REST APIs, Alamofire
AI Assistant	OpenAI API, Firebase NLP (or native prompts)
Charts	Swift Charts
Notifications	UserNotifications framework
Collaboration	GitHub (branching, pull requests)
Beta Testing	TestFlight