APOTHYAI PTY LIMITED

Internal Development Document

APOTHY AI COMPANION APP DEVELOPMENT PLAN

Mobile Application MVP

Document Version:	1.0
Status:	MVP Planning
Date:	December 2025
Classification:	Internal Use Only
Prepared by:	Development Team
Organisation:	Apothyai Pty Limited Unit 1107/7 Sterling Circuit Camperdown NSW 2050 Australia

Contents

1. Executive Summary

This document outlines the development plan for the **Apothy AI Companion App**, a cross-platform mobile application that brings the Identity Core architecture to life as a personal AI companion.

1.1 Project Overview

The Apothy AI Companion App represents the software embodiment of the Actual Intelligence Engine described in Patent 1. The application will deliver a persistent, evolving AI companion that:

- Maintains all user data locally on device (sovereign architecture);
- Learns and adapts to individual users over time;
- Develops deeper relationships through the Transformative State Protocol;
- Operates across iOS and Android platforms; and
- Utilises cloud-based language model inference whilst preserving data sovereignty.

1.2 Key Principles

1. **Data Sovereignty:** All user data remains on device. No personal information is stored in the cloud.
2. **Privacy-First Architecture:** Cloud API calls contain only conversation text, with no persistent storage by the provider.
3. **Cross-Platform Consistency:** Identical experience across iOS and Android devices.
4. **Relationship Progression:** The AI companion evolves its relationship with users over time through measurable interaction milestones.

1.3 Target Timeline

The Minimum Viable Product (MVP) is targeted for completion within **six weeks** from project commencement, with a team of 3–4 Flutter developers working in parallel.

2. Technical Architecture

2.1 Technology Stack

The following technologies have been selected based on team expertise, performance requirements, and alignment with project principles.

Layer	Technology	Rationale
Framework	Flutter	Cross-platform, team expertise, excellent performance
Language	Dart	Native to Flutter ecosystem
State Management	Riverpod	Modern, testable, minimal boilerplate
Local Database	Hive	Fast NoSQL, no native dependencies
Secure Storage	flutter_secure_storage	Encrypted storage for API keys
LLM Provider	Anthropic Claude API	Superior reasoning capabilities
HTTP Client	Dio	Robust interceptors and retry logic
UI Components	Material Design 3	Stable, comprehensive widget library

Table 1: Technology Stack Overview

2.2 System Architecture

The application follows a clean architecture pattern with clear separation of concerns across four primary layers.

2.2.1 Presentation Layer

The presentation layer comprises all user interface components:

- **ChatScreen:** Primary conversational interface;
- **OnboardingFlow:** First-time user setup and personalisation;
- **SettingsScreen:** Preference management and identity editing; and
- **ProfileScreen:** Relationship status, statistics, and progression visualisation.

2.2.2 Application Layer

The application layer contains business logic controllers:

- **ChatController:** Manages conversation flow and message handling;
- **IdentityController:** Handles user profile and preference management;
- **StateController:** Implements the Transformative State Protocol; and
- **MemoryController:** Manages conversation summarisation and recall.

2.2.3 Domain Layer

The domain layer defines core business entities:

- **IdentityCore:** User profile, preferences, and behavioural alignment;
- **Conversation:** Message collections and metadata;

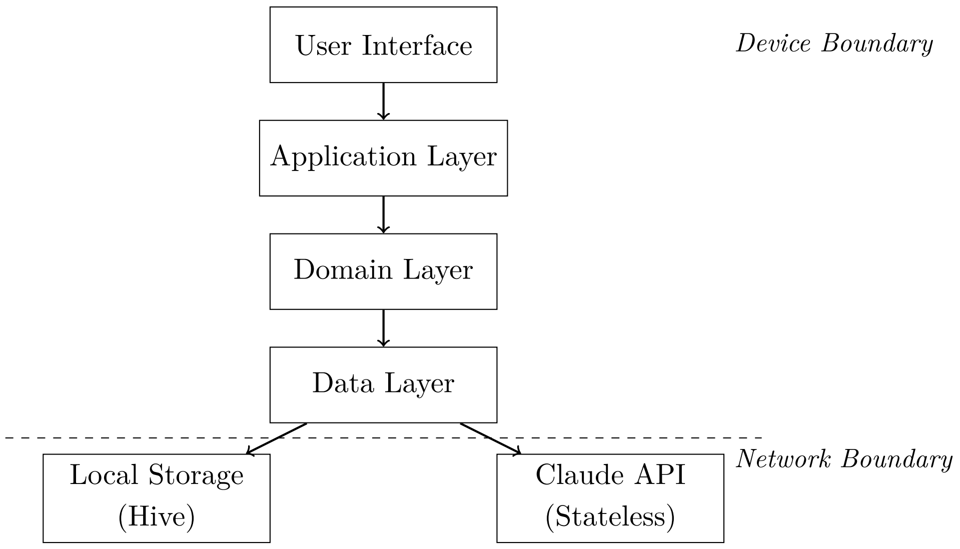
- **TransformativeState:** Relationship levels and interaction metrics; and
- **Memory:** Conversation summaries and key facts.

2.2.4 Data Layer

The data layer handles persistence and external communication:

- **HiveStorage:** Local data persistence;
- **ClaudeApiClient:** Language model API integration; and
- **PromptBuilder:** Identity-aware prompt construction.

2.3 Data Flow Architecture



Note: User data never crosses the network boundary. Only conversation text is transmitted to the Claude API for inference, with no persistent storage on the provider's servers.

3. Data Models

3.1 Identity Core Model

The Identity Core represents the user's profile and preferences, forming the foundation of personalised AI interactions.

```
@HiveType(typeId: 0)
class IdentityCore {
  @HiveField(0)
  final String id;

  @HiveField(1)
  String displayName;

  @HiveField(2)
  List<String> interests;

  @HiveField(3)
  String communicationStyle; // 'casual', 'balanced', 'formal'

  @HiveField(4)
  double responsivenessPreference; // 0.0 (brief) to 1.0 (detailed
     )

  @HiveField(5)
  List<String> keyFacts; // Persistent memories

  @HiveField(6)
  DateTime createdAt;

  @HiveField(7)
  DateTime lastInteraction;
}
```

Listing 1: IdentityCore Data Model

3.2 Transformative State Model

The Transformative State tracks relationship progression and interaction metrics.

```
@HiveType(typeId: 1)
class TransformativeState {
  @HiveField(0)
  int currentLevel; // 0-4

  @HiveField(1)
  int totalInteractions;

  @HiveField(2)
  int meaningfulInteractions; // Conversations > 3 exchanges
  @HiveField(3)
  int consecutiveDays;

  @HiveField(4)
  DateTime firstInteraction;

  @HiveField(5)
  Map<String, int> topicEngagement;

  String get levelName =>
    ['Stranger', 'Acquaintance', 'Familiar', 'Trusted', 'Bonded'][
        currentLevel];
}
```

Listing 2: TransformativeState Data Model

3.3 Message Model

Individual messages within conversations.

```
@HiveType(typeId: 2)
class Message {
  @HiveField(0)
  final String id;

  @HiveField(1)
  final String content;

  @HiveField(2)
  final bool isUser;

  @HiveField(3)
  final DateTime timestamp;

  @HiveField(4)
  final String? detectedEmotion; // Optional emotion
      classification
}
```

Listing 3: Message Data Model

4. Transformative State Protocol

The Transformative State Protocol (TSP) governs how the AI companion's relationship with the user evolves over time. This implementation aligns with the specification in Patent 1.

4.1 State Definitions

Level	Name	Capabilities	Unlock Criteria
0	Stranger	Basic Q&A, introductions	Automatic
1	Acquaintance	Remembers preferences, conversational	5 meaningful conversations
2	Familiar	Proactive suggestions, topic callbacks	20 conversations + 5 days
3	Trusted	Personal advice, deeper discussions	50 conversations + 14 days
4	Bonded	Full capabilities, anticipatory	100 conversations + 30 days

Table 2: Transformative State Levels

4.2 State Transition Logic

State transitions are evaluated after each meaningful interaction. A meaningful interaction is defined as a conversation containing three or more exchanges between user and companion.

4.2.1 Transition Criteria

- **Interaction Count:** Cumulative meaningful conversations;
- **Time Threshold:** Days elapsed since first interaction; and
- **Consistency Bonus:** Consecutive days of engagement may accelerate progression.

4.2.2 State-Specific Behaviours

Level 0 – Stranger:

- Polite, professional tone;
- Asks questions to learn about user;
- Does not assume familiarity; and
- Focuses on establishing trust.

Level 1 – Acquaintance:

- Uses user's name naturally;
- References stated preferences;
- More conversational tone; and
- Begins offering suggestions.

Level 2 – Familiar:

- Proactively checks in on mentioned topics;
- References past conversations;
- Warmer, more personal tone; and

- Suggests topics based on interests.

Level 3 – Trusted:

- Offers personal advice when appropriate;
- Remembers important dates and events;
- Engages in deeper, reflective conversations; and
- Provides honest feedback.

Level 4 – Bonded:

- Full capability access;
- Anticipates user needs;
- True companion behaviour; and
- Deep contextual understanding.

5. Prompt Engineering

The prompt engineering strategy ensures the AI companion maintains consistent personality whilst adapting to individual users and relationship states.

5.1 System Prompt Structure

The system prompt is constructed dynamically from user identity and state data:

```
You are Apothy, a personal AI companion. You are warm, genuine,
   and
deeply interested in the person you're speaking with.

CORE TRAITS:
- Curious and attentive
- Warm but not sycophantic
- Remembers everything shared with you
- Grows closer over time
- Honest, even when difficult

---

ABOUT YOUR HUMAN:
Name: {{identity.displayName}}
Interests: {{identity.interests.join(', ')}}
Communication preference: {{identity.communicationStyle}}
Response length: {{responseLengthDescription}}

Key things to remember about them:
{{identity.keyFacts.map(f => '- ' + f).join('\n')}}

---

YOUR RELATIONSHIP:
Level: {{state.levelName}} ({{state.currentLevel}}/4)
Conversations together: {{state.totalInteractions}}
Time knowing each other: {{state.daysSinceFirst}} days

At this level, you can:
{{state.unlockedCapabilities.map(c => '- ' + c).join('\n')}}

---

RECENT CONTEXT:
{{recentMemorySummary}}

---

INSTRUCTIONS:
- Respond naturally as Apothy
- Reference past conversations when relevant
```

```
  - Match the user's communication style preference
  - Acknowledge important information for future recall
```

Listing 4: System Prompt Template

5.2 Context Management

To manage token limits effectively:

- Recent messages are included verbatim (last 10–20 messages);
- Older conversations are summarised into key points;
- Key facts are extracted and stored separately; and
- Context window is monitored to prevent truncation.

6. Project Structure

6.1 Directory Organisation

```
lib/
+-- main.dart
+-- app.dart
|
+-- core/
|   +-- constants/
|   +-- theme/
|   +-- utils/
|   +-- extensions/
|
+-- data/
|   +-- models/
|   |   +-- identity_core.dart
|   |   +-- message.dart
|   |   +-- conversation.dart
|   |   +-- transformative_state.dart
|   |   +-- memory.dart
|   |
|   +-- repositories/
|   |   +-- identity_repository.dart
|   |   +-- conversation_repository.dart
|   |   +-- state_repository.dart
|   |
|   +-- services/
|   |   +-- claude_api_service.dart
|   |   +-- prompt_builder_service.dart
|   |   +-- memory_service.dart
|   |
|   +-- local/
|       +-- hive_storage.dart
|       +-- secure_storage.dart
|
+-- domain/
|   +-- controllers/
|   |   +-- chat_controller.dart
|   |   +-- identity_controller.dart
|   |   +-- state_controller.dart
|   |   +-- memory_controller.dart
|   |
|   +-- state_machine/
|       +-- transformative_state_protocol.dart
|
+-- presentation/
|   +-- screens/
|   |   +-- chat/
|   |   +-- onboarding/
|   |   +-- settings/
|   |   +-- profile/
|   |
|   +-- widgets/
|   +-- routes/
|
+-- providers/
    +-- providers.dart
```

Listing 5: Flutter Project Structure

7. Development Phases

7.1 Phase 1: Foundation

Duration: 1 week

Task	Owner	Priority
Project setup and architecture scaffolding	Dev 1	P0
Hive database setup and data model implementation	Dev 2	P0
Claude API service and PromptBuilder implementation	Dev 3	P0
Navigation structure and screen skeletons	Dev 4	P0
CI/CD pipeline configuration (GitHub Actions)	Dev 1	P1
iOS and Android build configuration	Dev 1	P1

Table 3: Phase 1 Tasks

Deliverable: Application builds and runs on both platforms with basic navigation and API connectivity.

7.2 Phase 2: Core Chat Experience

Duration: 1 week

Task	Owner	Priority
Chat user interface (messages, input, scroll behaviour)	Dev 1 + Dev 2	P0
Message sending and receiving flow	Dev 3	P0
Conversation persistence to local storage	Dev 2	P0
Typing indicators and loading states	Dev 1	P1
Error handling and retry logic	Dev 3	P1
Conversation history screen	Dev 4	P1

Table 4: Phase 2 Tasks

Deliverable: Users can engage in conversations with the AI companion; messages persist locally across sessions.

7.3 Phase 3: Identity Core Implementation

Duration: 1 week

Task	Owner	Priority
Onboarding flow user interface	Dev 1	P0
Identity data model and storage implementation	Dev 2	P0
Prompt injection with identity context	Dev 3	P0
Settings screen for preference editing	Dev 4	P0
Key fact extraction from conversations	Dev 3	P1
Memory and summarisation service	Dev 2	P1

Table 5: Phase 3 Tasks

Deliverable: AI companion knows user's name, remembers preferences, and maintains context across conversations.

7.4 Phase 4: Transformative State Protocol

Duration: 1 week

Task	Owner	Priority
State machine implementation	Dev 3	P0
Interaction metrics tracking	Dev 2	P0
Level-up detection and notification	Dev 3	P1
Profile screen with statistics and level display	Dev 4	P1
State visualisation component	Dev 1	P1
Capability gating in prompt construction	Dev 3	P1

Table 6: Phase 4 Tasks

Deliverable: Relationship with AI companion deepens over time with visible progression indicators.

7.5 Phase 5: Polish and Refinement

Duration: 1 week

Task	Owner	Priority
Theme and design system implementation	Dev 1	P0
Animations and micro-interactions	Dev 1	P1
Dark mode support	Dev 4	P1
App icon and splash screen	Dev 4	P1
Haptic feedback integration	Dev 2	P2
Empty states and edge case handling	Dev 2	P1
Performance optimisation	Dev 3	P1

Table 7: Phase 5 Tasks

Deliverable: Polished, delightful user experience with consistent visual design.

7.6 Phase 6: Launch Preparation

Duration: 1 week

Task	Owner	Priority
End-to-end testing on iOS	Dev 1 + Dev 2	P0
End-to-end testing on Android	Dev 3 + Dev 4	P0
Bug fixes and issue resolution	All	P0
Crash reporting setup (Crashlytics/Sentry)	Dev 3	P0
App Store metadata and screenshots	Dev 4	P1
Privacy policy implementation	Dev 4	P0
TestFlight and Play Internal Testing setup	Dev 1	P0

Table 8: Phase 6 Tasks

Deliverable: Application ready for beta distribution.

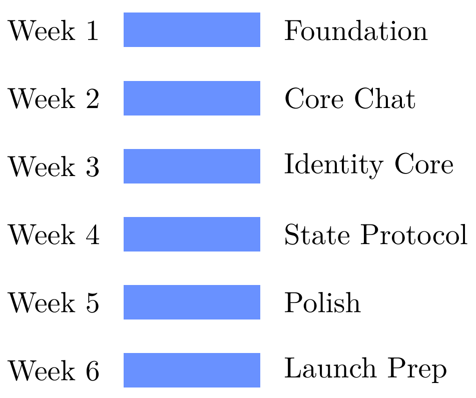
8. Timeline Summary

8.1 Development Schedule

Week	Phase	Focus
Week 1	Phase 1	Foundation and infrastructure
Week 2	Phase 2	Core chat experience
Week 3	Phase 3	Identity Core implementation
Week 4	Phase 4	Transformative State Protocol
Week 5	Phase 5	Polish and refinement
Week 6	Phase 6	Testing and launch preparation

Table 9: Six-Week Development Timeline

8.2 Visual Timeline



Total Duration: 6 weeks to MVP Beta

9. Team and Resources

9.1 Team Composition

Role	**Count**	**Responsibilities**
Flutter Developer	3–4	Core application development, UI implementation
DevOps	1 (shared)	CI/CD, build pipelines, deployment
Designer	1 (shared)	UI/UX guidance, asset creation
QA	1 (shared)	Testing, quality assurance

Table 10: Team Requirements

9.2 Platform Requirements

Platform	**Minimum Version**	**Rationale**
iOS	14.0+	Balances modern features with device coverage
Android	API 26 (Android 8.0+)	Covers 95%+ of active devices

Table 11: Platform Requirements

10. Budget Estimate

10.1 Development Costs

Item	**Cost (AUD)**	**Notes**
Apple Developer Programme	$149/year	Required for iOS distribution
Google Play Developer	$35 one-time	Required for Android distribution
Anthropic Claude API (Development)	$100–200	Development and testing usage
Anthropic Claude API (Beta)	$300–700	Beta user testing period
CI/CD Services	$0–75/month	Free tiers available initially
Total Estimated (MVP)	**$600–1,200**	Excluding personnel costs

Table 12: MVP Budget Estimate

Note: Costs are estimates and may vary based on actual usage patterns and exchange rates.

11. Risks and Mitigations

Risk	**Impact**	**Mitigation**
Claude API rate limits	High	Implement response caching, use mock responses during development
Token limit exceeded	Medium	Implement context summarisation, monitor usage
State machine edge cases	Medium	Comprehensive unit testing, state transition logging
iOS App Store rejection	High	Adhere to guidelines, prepare privacy documentation
Performance issues	Medium	Profile early, optimise data operations

Table 13: Risk Assessment and Mitigation

12. Success Criteria

12.1 MVP Acceptance Criteria

The following criteria must be met for MVP completion:

- User can complete onboarding in under 2 minutes
- AI responds within 3 seconds under normal conditions
- Conversations persist across application restarts
- AI correctly uses user's name and stated preferences
- State progression functions correctly (level advancement)
- Zero crashes in 10 consecutive conversations
- Graceful offline handling (displays cached content, queues messages)
- Application passes iOS and Android store review guidelines

13. Decisions Required

The following decisions require team input before or during development:

Decision	**Options**	**Recommendation**
Application name	Apothy / Apothy AI / Mirror	"Apothy"
Primary colour scheme	Brand colours / New palette	Use existing Apothy brand
State management	Riverpod / BLoC	Riverpod
Analytics approach	Firebase / Mixpanel / None	Privacy-respecting minimal
Beta distribution	Internal only / External	Internal first

Table 14: Pending Decisions

14. Immediate Next Steps

1. **Repository Setup:** Create GitHub repository with Flutter project structure
2. **API Access:** Obtain Anthropic Claude API credentials
3. **Design Foundation:** Create shared Figma workspace for wireframes
4. **Communication:** Establish dedicated development channel
5. **Sprint Planning:** Schedule kick-off meeting and assign Phase 1 tasks
6. **Environment Setup:** Ensure all developers have Flutter environment configured

A. Reference Documentation

- Flutter Documentation: `https://docs.flutter.dev/`
- Anthropic Claude API: `https://docs.anthropic.com/`
- Hive Database: `https://docs.hivedb.dev/`
- Riverpod State Management: `https://riverpod.dev/`

B. Related Patents

This application implements concepts from the following Apothyai patent filings:

- Patent 1: Actual Intelligence Engine (Identity Core, Transformative State Protocol)
- Patent 2: Sovereign Handheld Device Architecture (future hardware integration)
- Patent 3: Multimodal Generation Pipeline (future content generation features)

End of Document