EaseMind

Contributors:

Suhaas Srungavarapu

Saksham Yadav

Dhruvi Kothari

Table Of Contents

Introduction:

The Emotional Calming Mobile App continuously monitors a user's emotional signals—SMS content, email content, and manual chat-surveys—to detect anxiety or panic attacks. When elevated stress is detected, it delivers personalized, psychology-backed interventions (guided breathing, grounding exercises, supportive messages) via in-app UI, push notifications, or text-to-speech.

Problem Statement:

Users suffering from anxiety or panic often lack immediate, personalized calming support when they need it most. Reliance on manual coping strategies (apps, helplines) can be slow or generic.

Objectives:

- Real-Time Detection of emotional distress via SMS, email, and optional survey/chatbot input.
- Personalized Intervention using an LLM agent (via OpenRouter) to craft calming messages or audio guidance.
- Modular Architecture leveraging design patterns for future extensibility (wearables integration, additional signals).

Requirement Analysis:

Function Requirements:

SMS Handler: Ingest incoming SMS text and compute panic score.

Email Handler: Ingest email body text for panic analysis.

Manual Input Handler: Accept user mood survey or chatbot conversation.

Panic Detector: Use an LLM (via OpenRouter) to return a 0.0–1.0 panic score.

Intervention Service: Build prompt, call LLM for calming-text, and return to caller.

REST API: Expose a single POST /api/panic-intervention endpoint.

Non-functional Requirements:

Performance: < 500 ms end-to-end response for typical prompt size.

Reliability: Gracefully fallback to mock LLM if API is unavailable.

Security: API key for OpenRouter must be injected via environment variable.

Testability: Automated tests exercising extreme‑case curl scenarios.

User Stories:

1. As a user, I want my incoming SMS monitored for distress so I can get help automatically.
    - Acceptance Criteria:
        - POST with "sms":"..." returns panicScore > 0 and calmingText.

2. As a user, I want my incoming email monitored for distress so I can get help automatically.
    - Acceptance Criteria:
        - POST with "email":"..." returns panicScore > 0 and calmingText.

3. As a user, I want to fill out a mood survey or chat with a bot so I can get personalized support.
    - Acceptance Criteria:
        - POST with "mood":"...","surveyAnswers":[...] returns calm advice.

4. As a developer, I want clear error responses for malformed payloads so I can handle client errors.
    - Acceptance Criteria:
        - POST with missing/invalid fields returns 400 Bad Request.

UML Diagram:

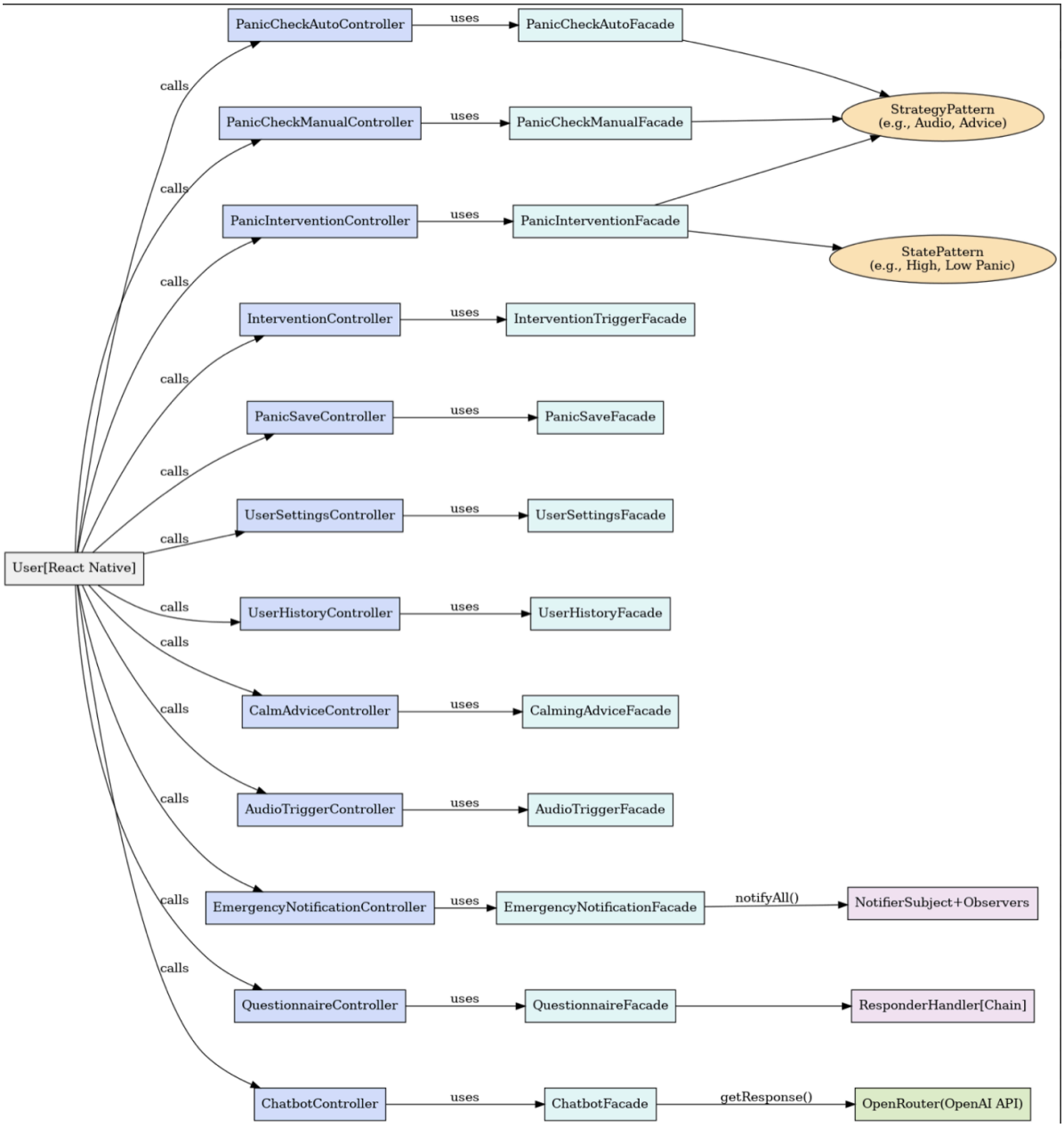


Figure 1: Integrated System UML

Activity Diagram:

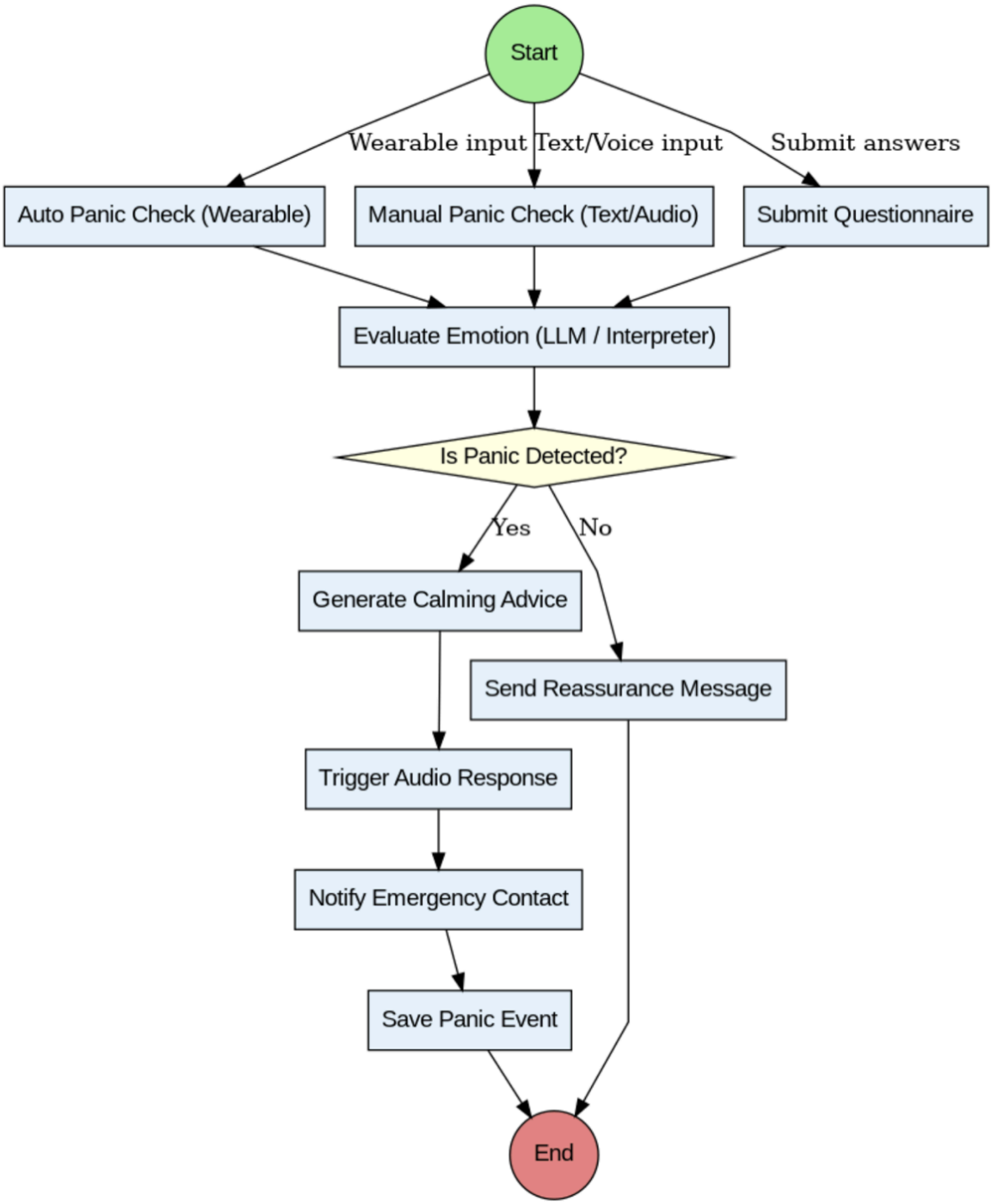


Figure 2: Integrated Panic Flow

Class Diagrams:

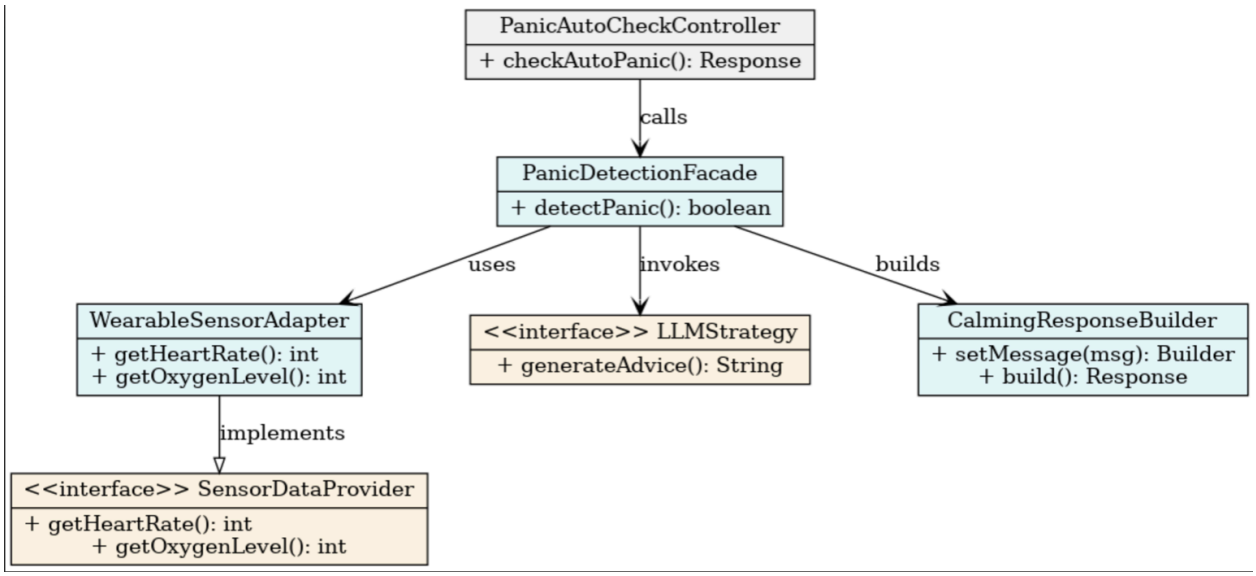


Figure 3: API_1 - Auto Panic check

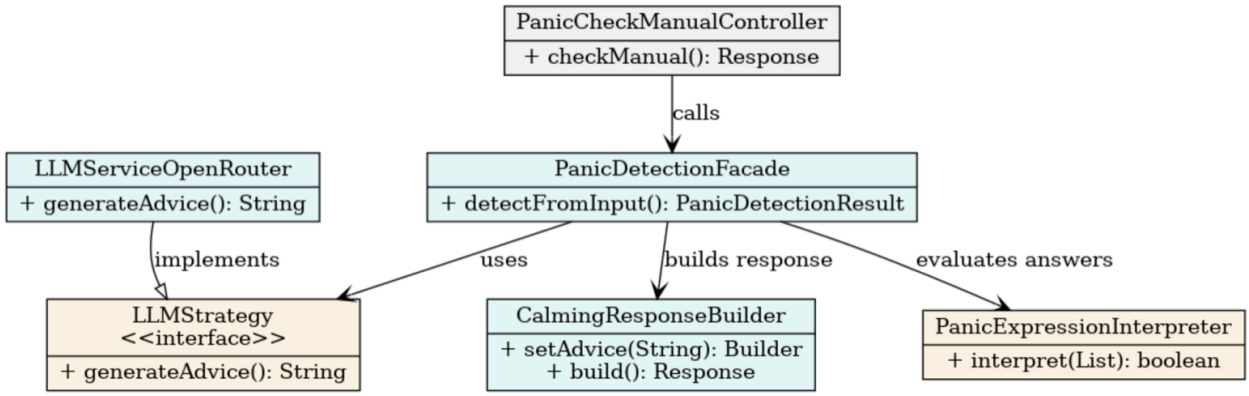


Figure 4: API_2 - Panic Check Manual

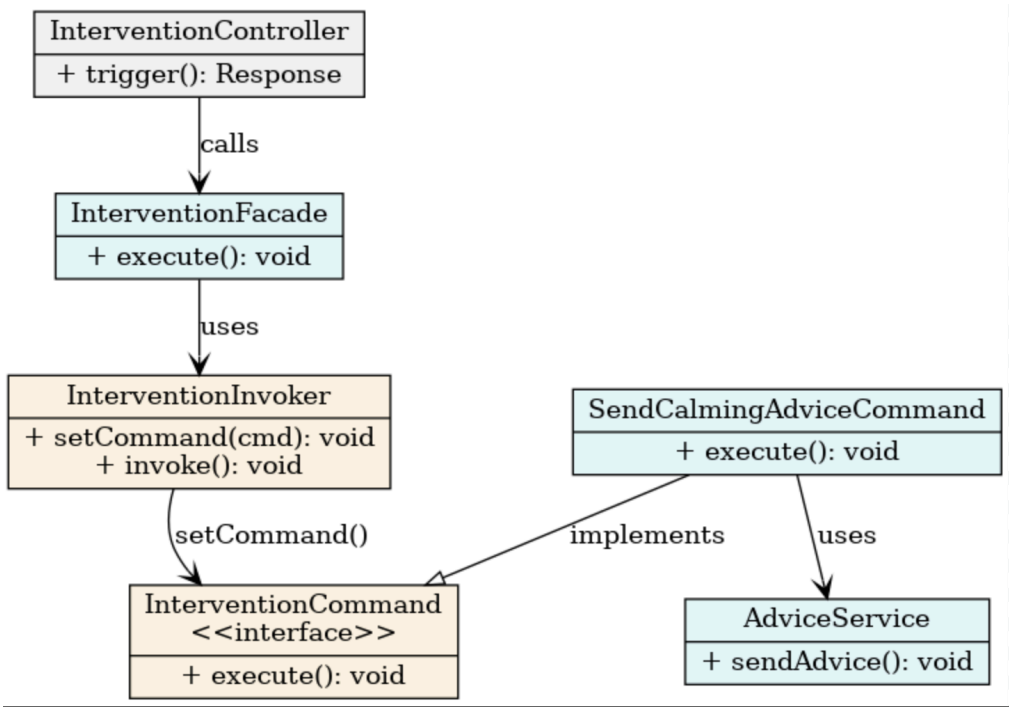


Figure 5: API_3 - Intervention Trigger

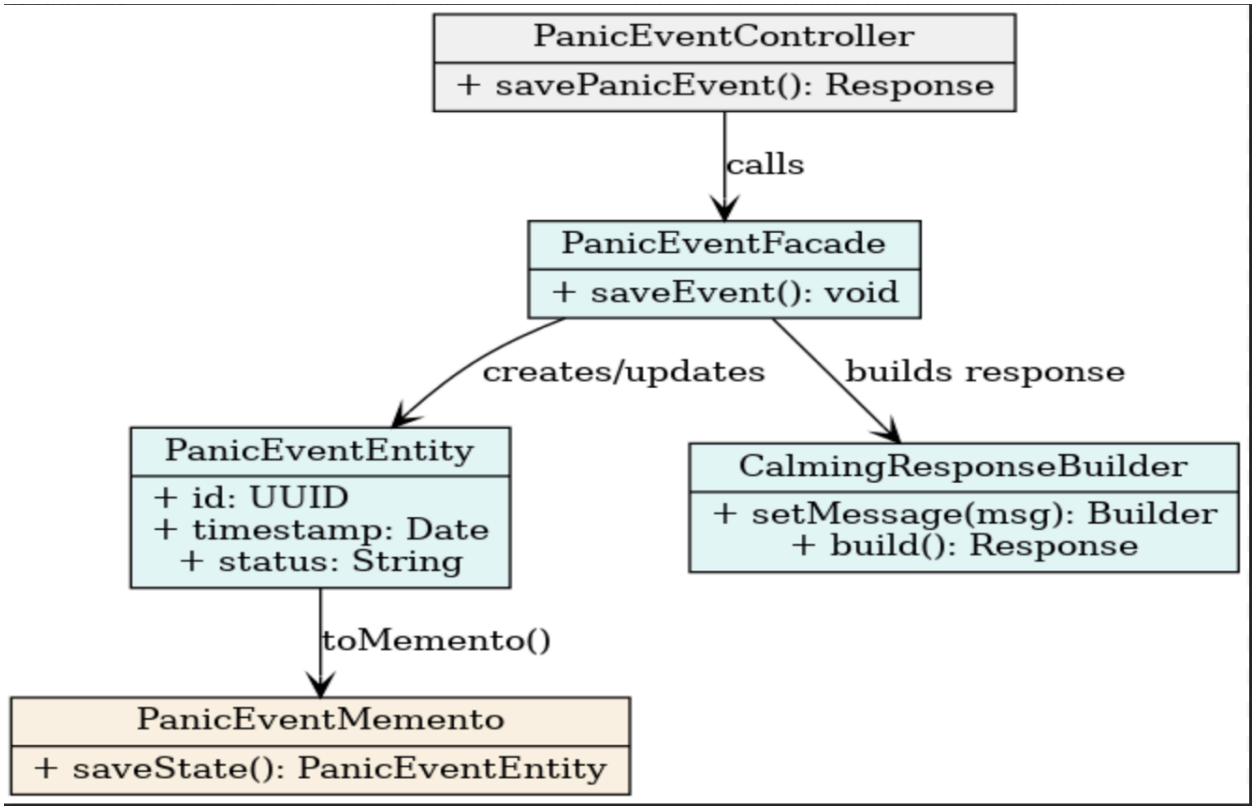


Figure 6: API_4 - Panic Save

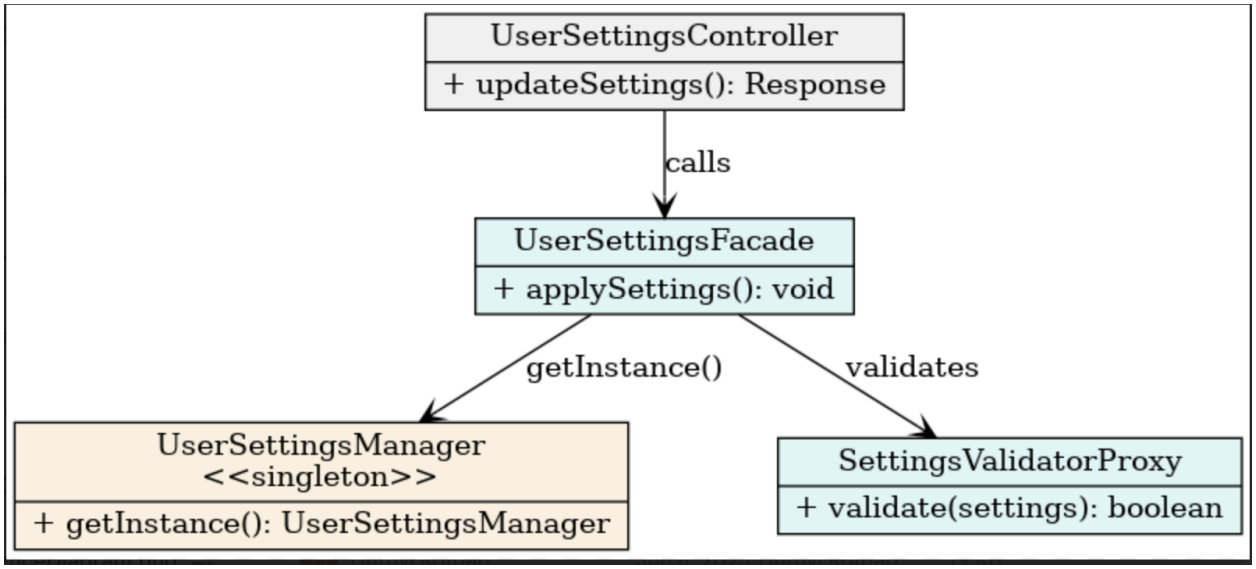


Figure 7: API_5 - User Settings

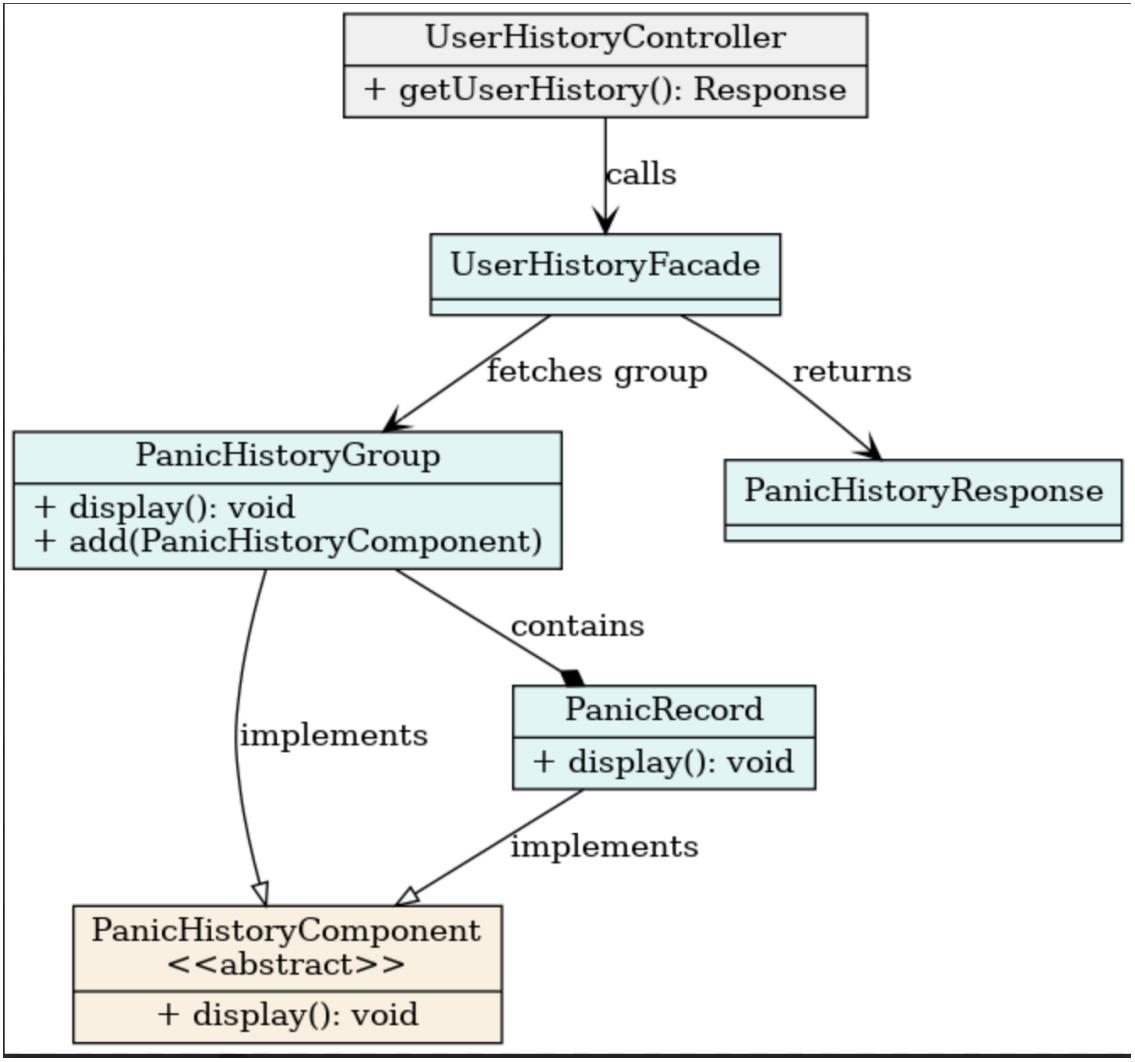


Figure 8: API_6 - User History

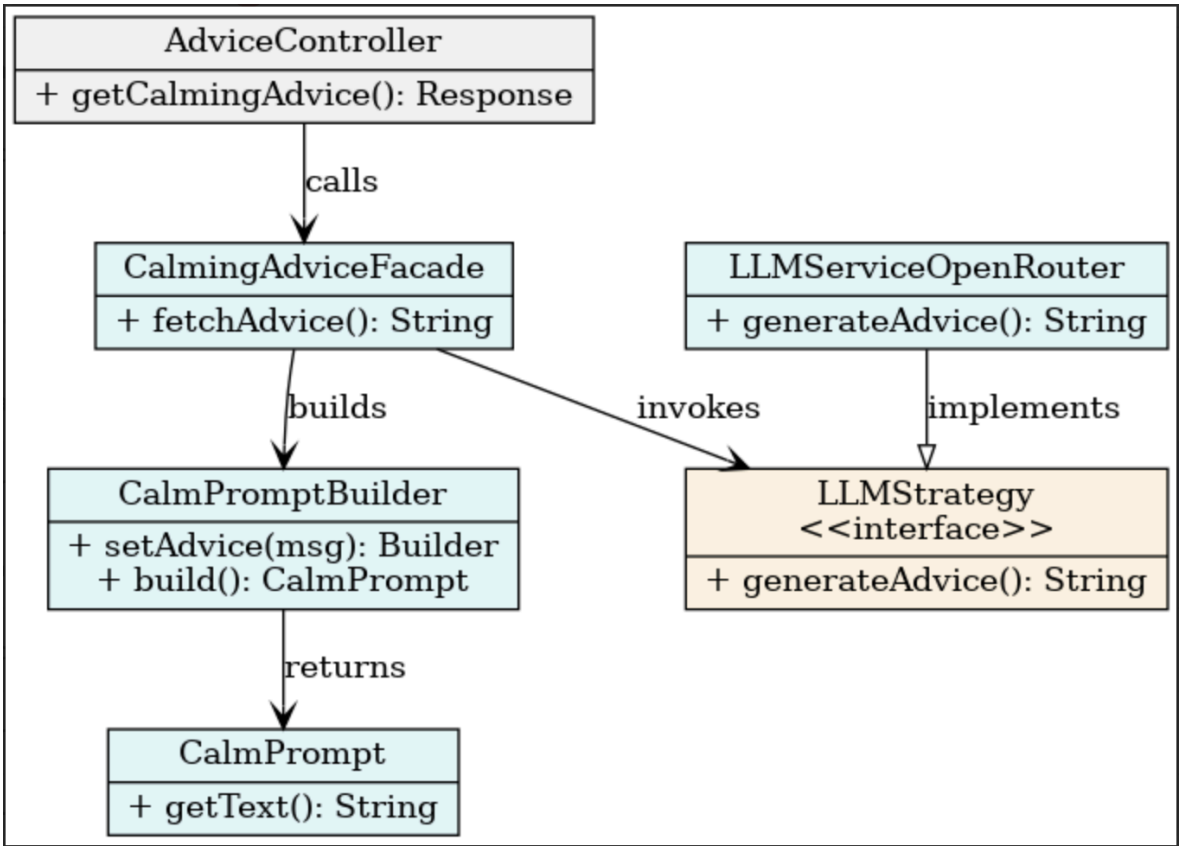


Figure 9: API_7 - Calming Advice

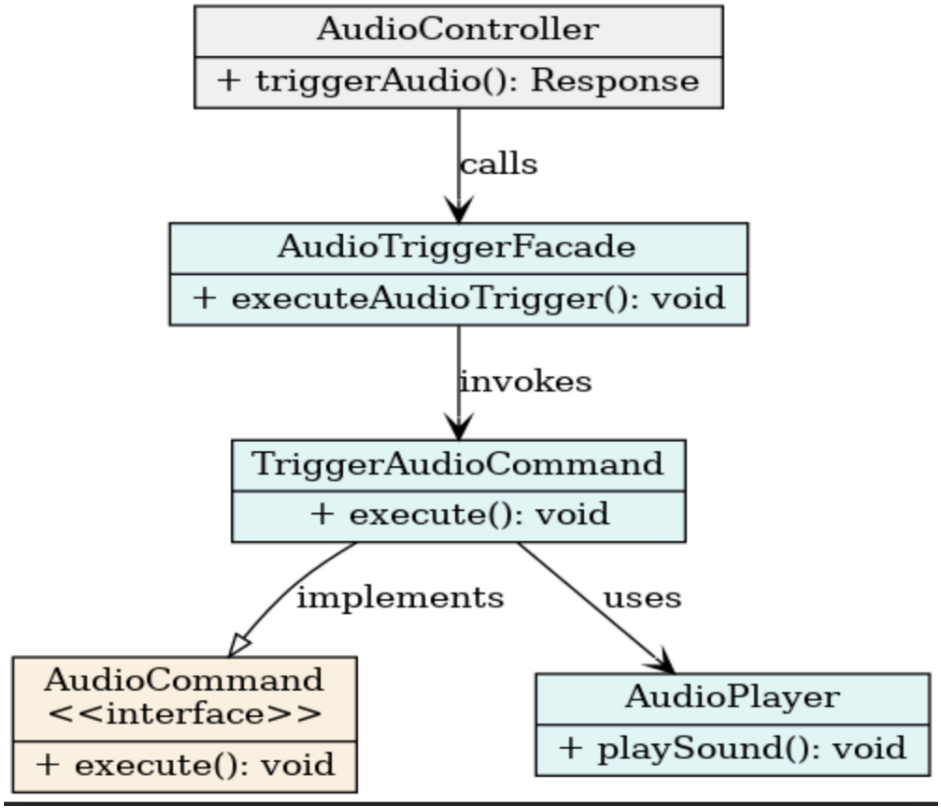


Figure 10: API_8 - Audio Trigger

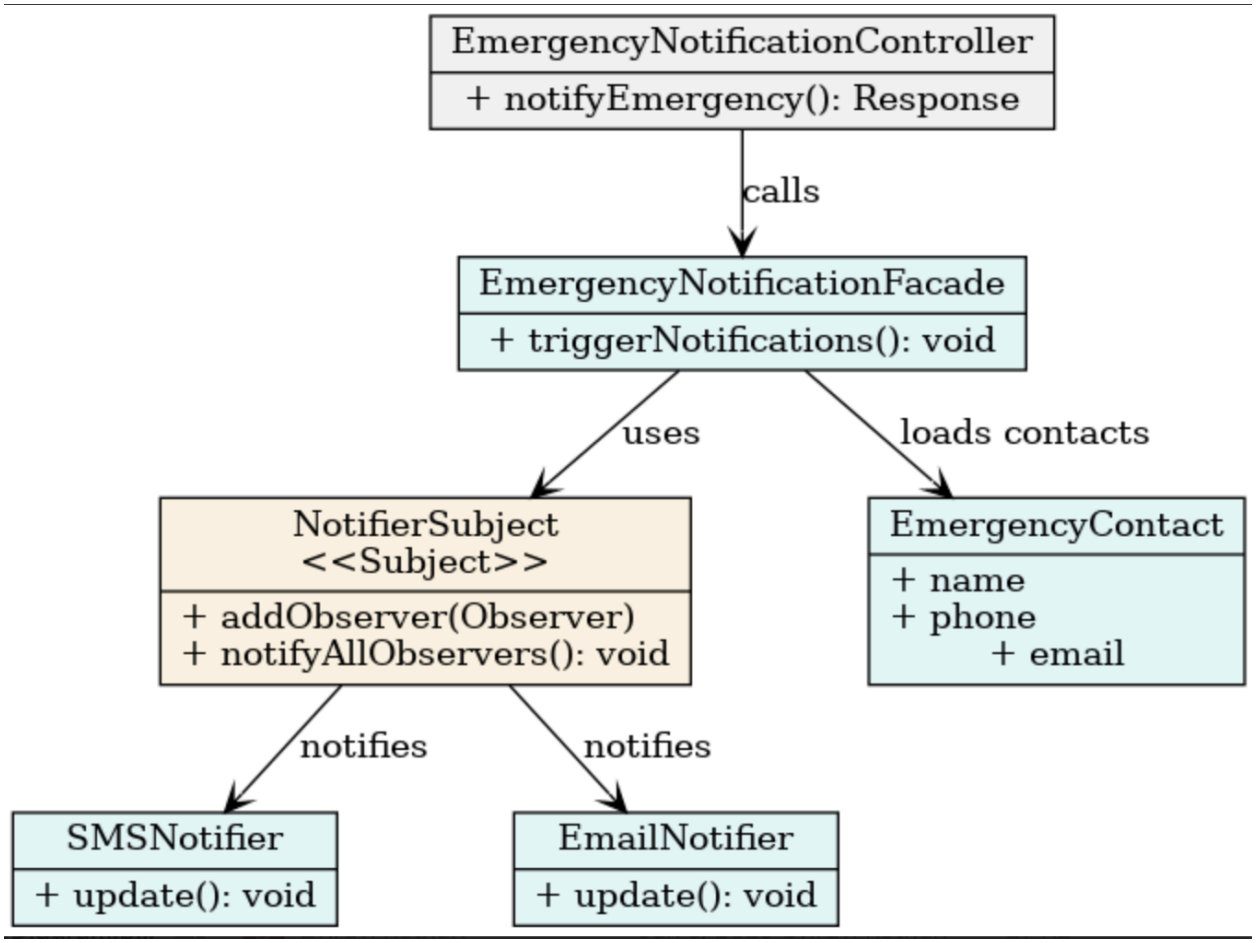


Figure 11: API_9 - Notify Emergency

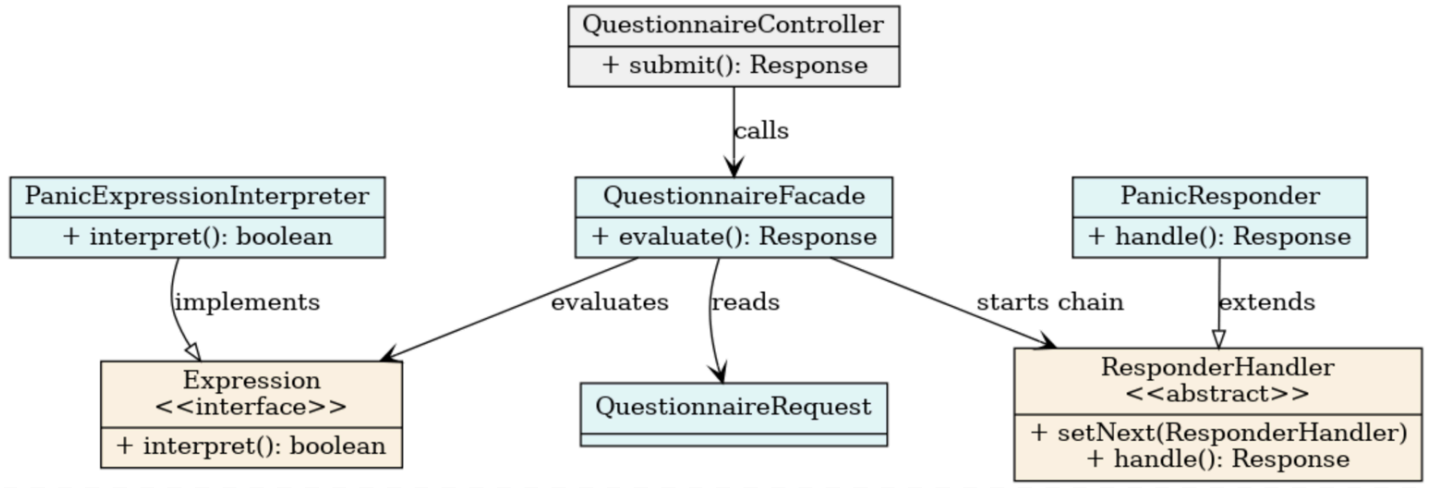


Figure 12: API_10 - Questionnaire Submit

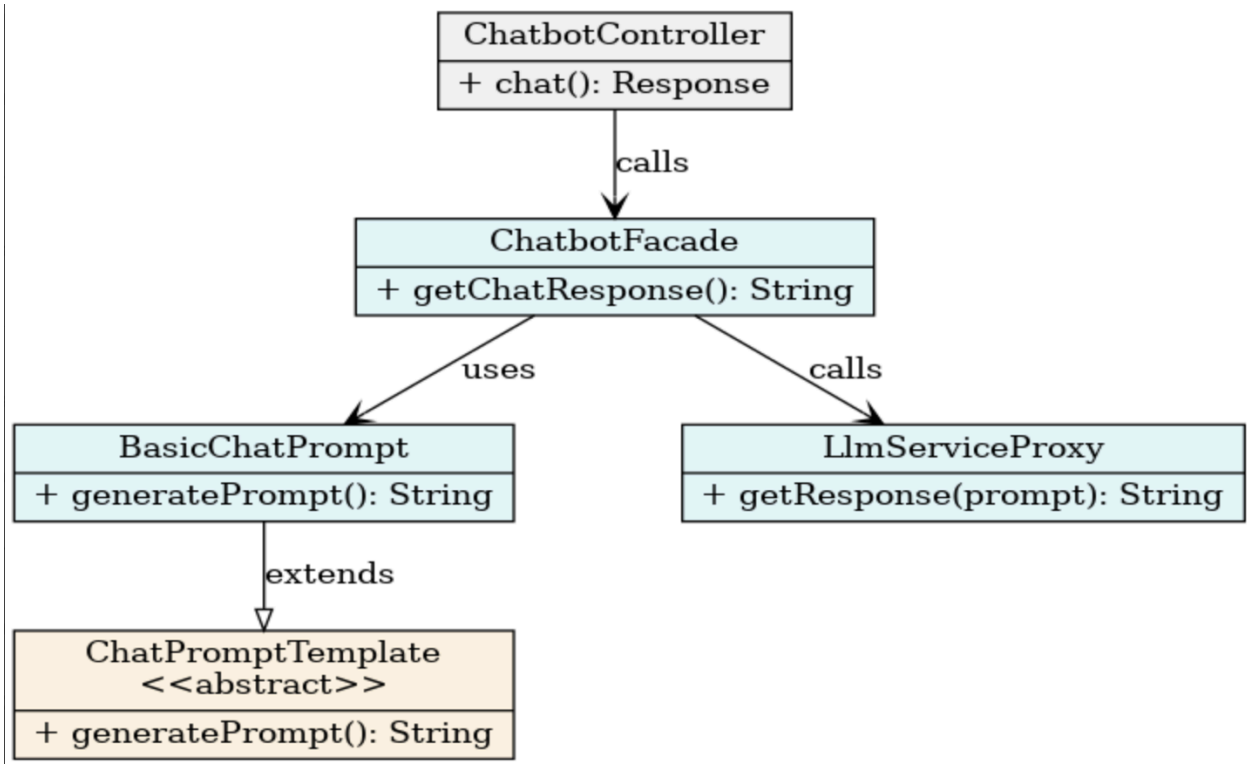


Figure 13: API_11 – Chatbot

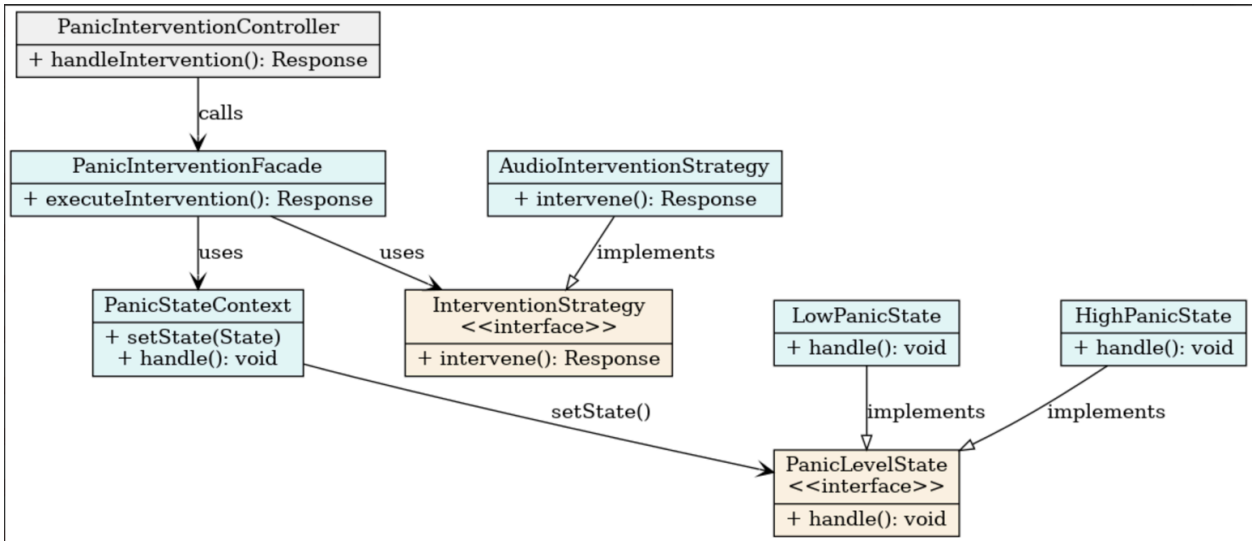


Figure 14: API_12 - Panic Intervention

Use case Diagrams:

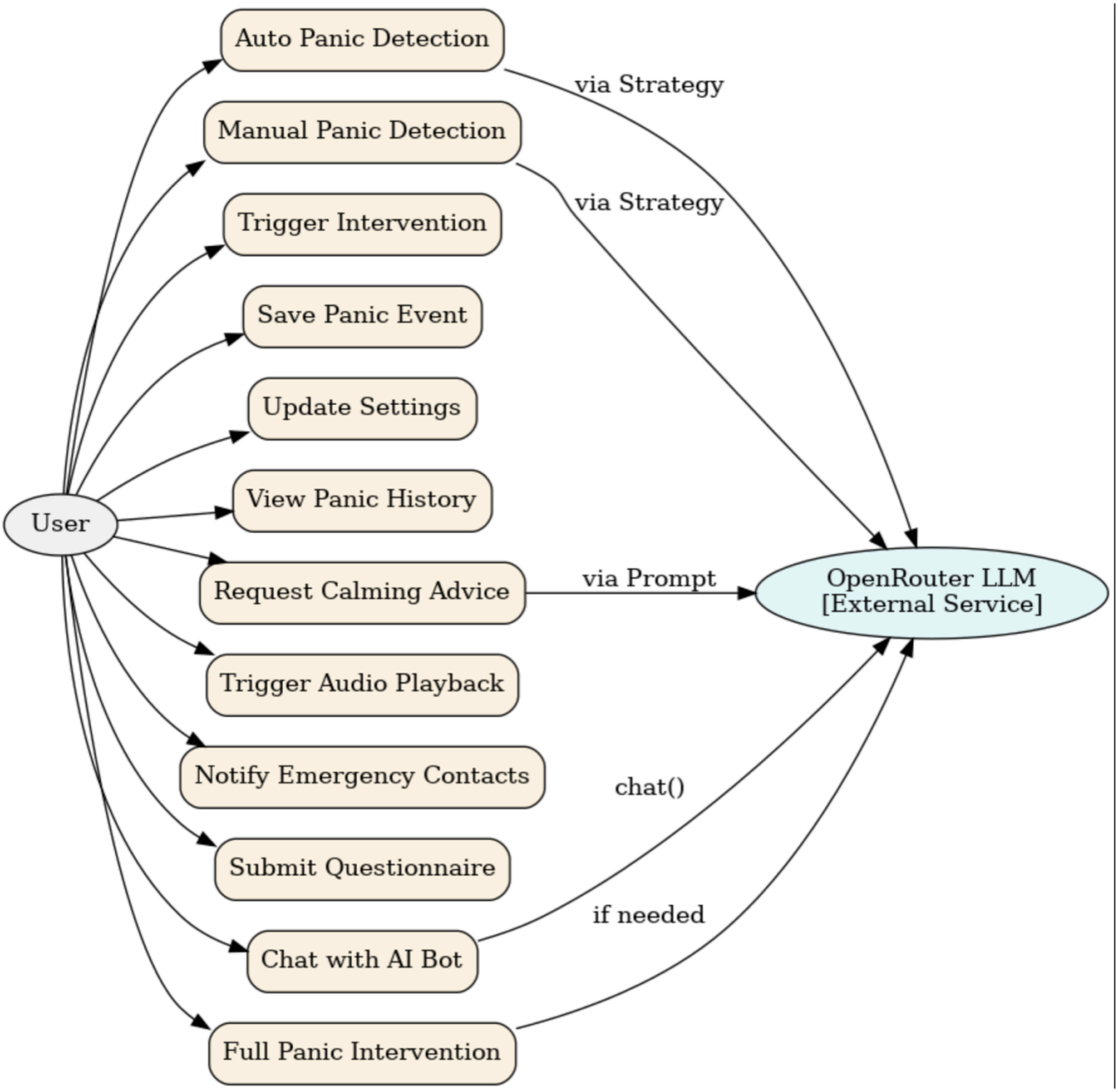


Figure 15: Use case overview

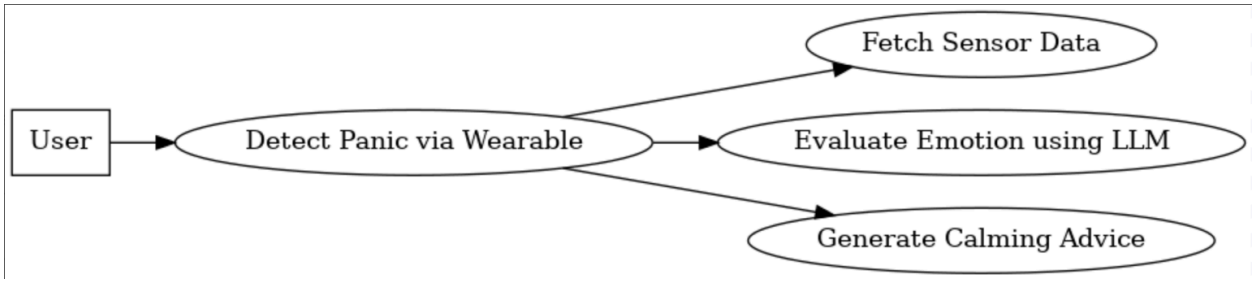


Figure 16: Auto Panic Detection

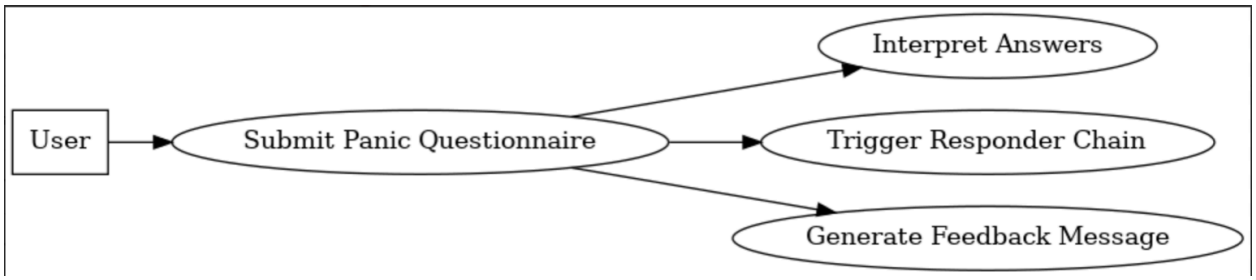


Figure 17: Manual Panic Check

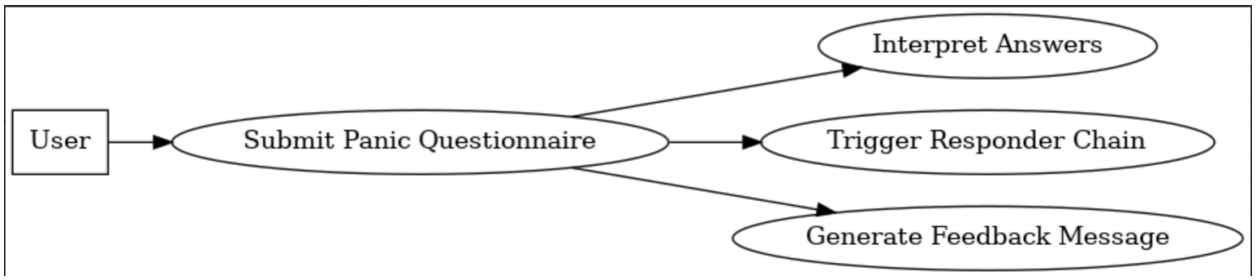


Figure 18: Questionnaire submit

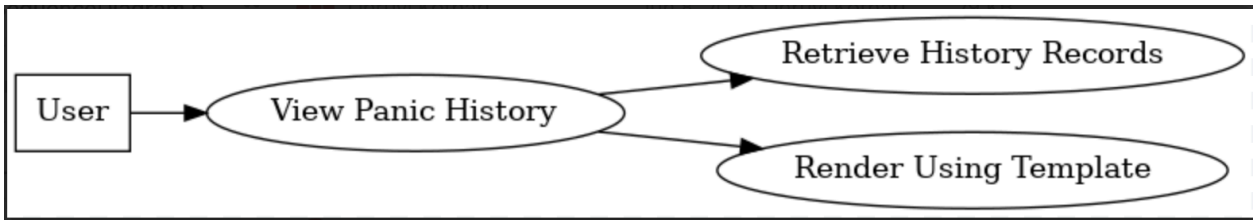


Figure 19: User History

Object Diagram:

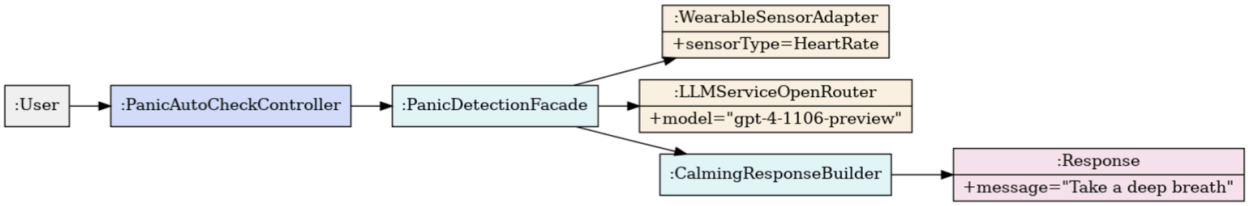


Figure 20: API_1

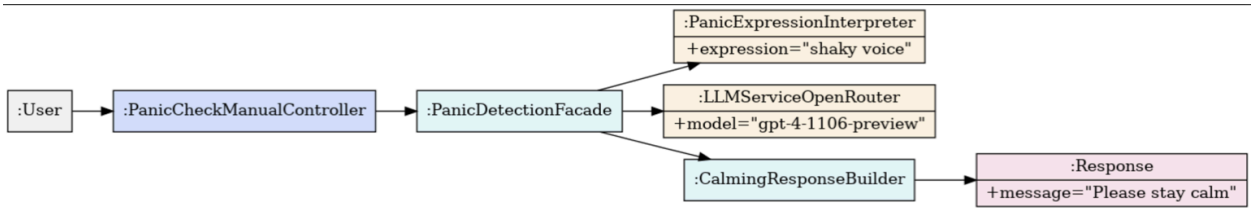


Figure 21: API_2

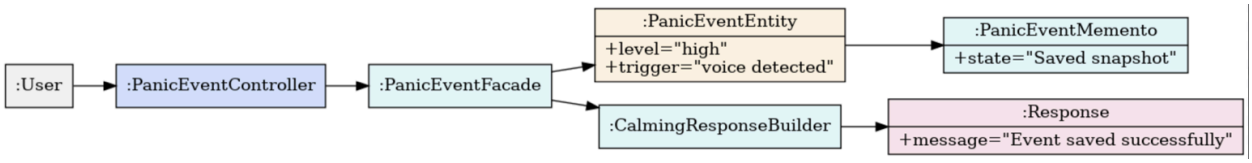


Figure 22: API_4

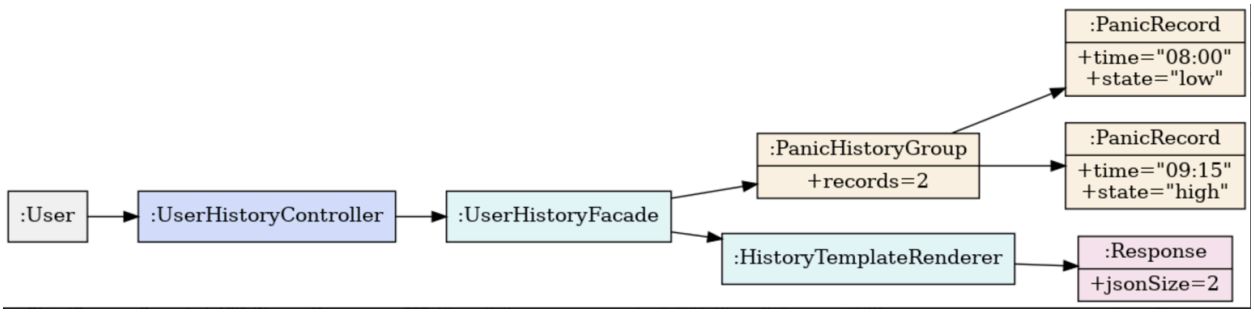


Figure 23: API_6

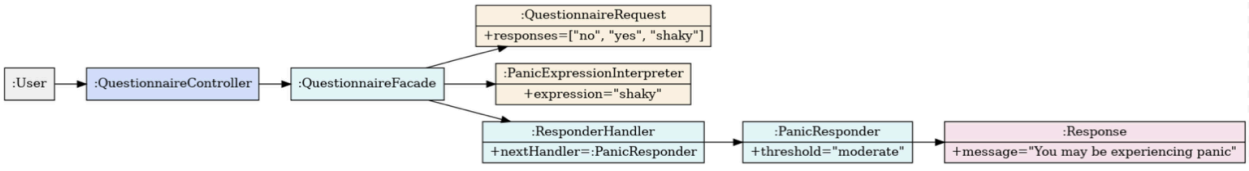


Figure 24: API_10

Sequence Diagram:

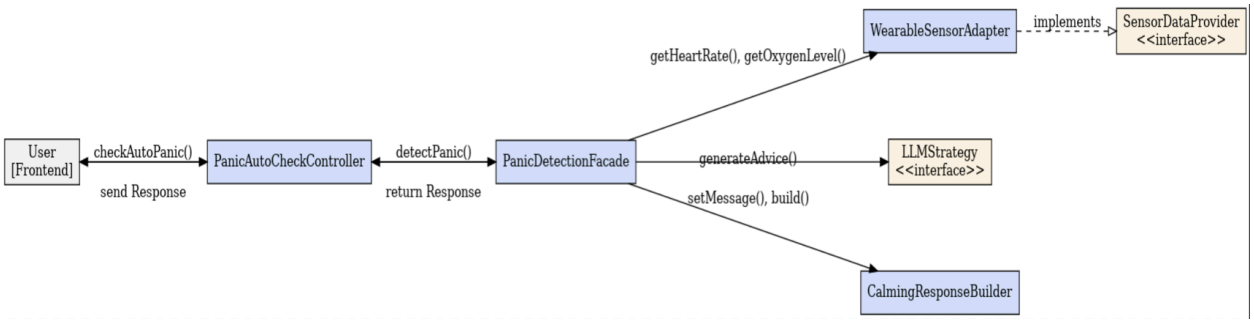


Figure 25: API_1 - Auto Panic check

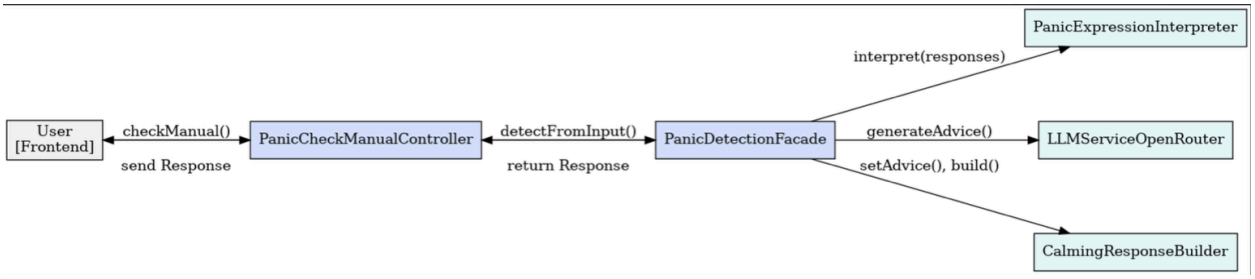


Figure 26: API_2 - Panic Check Manual

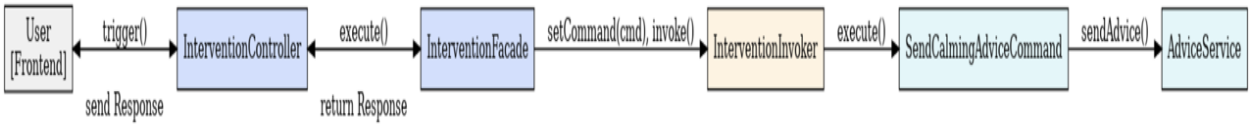


Figure 27: API_3 - Intervention Trigger

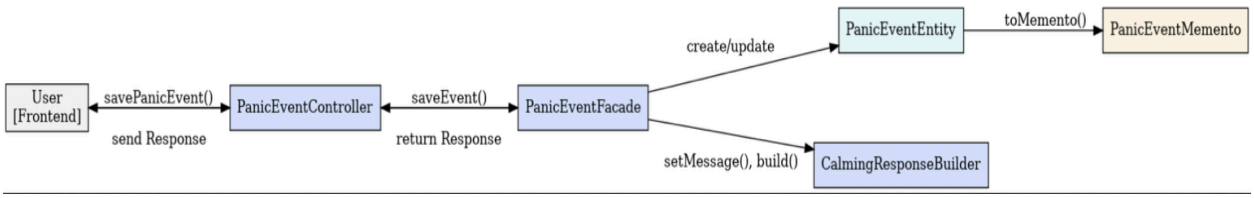


Figure 28: API_4 - Panic Save

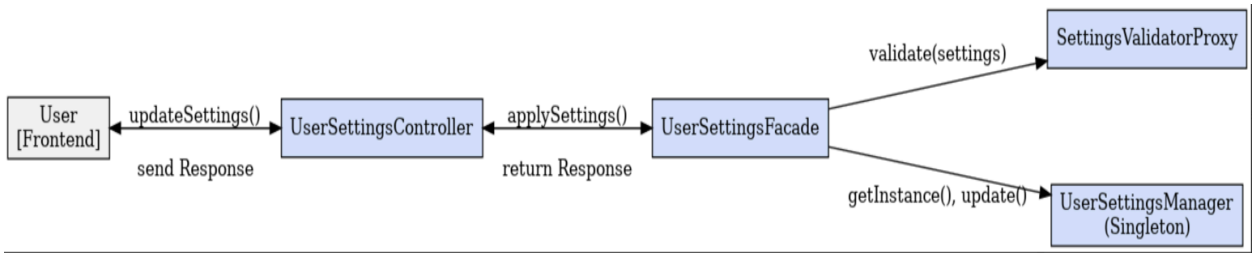


Figure 29: API_5 - User settings

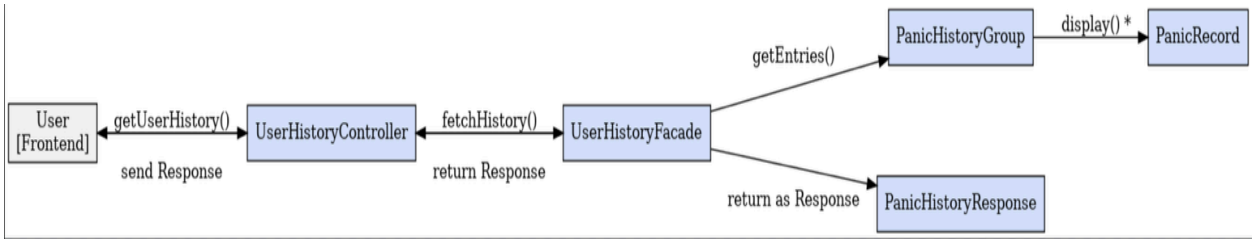


Figure 30: API_6 - User History

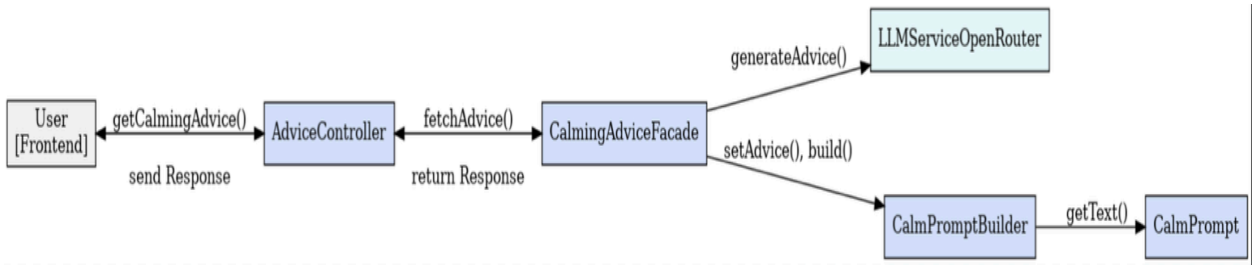


Figure 31: API_7 - Calming Advice

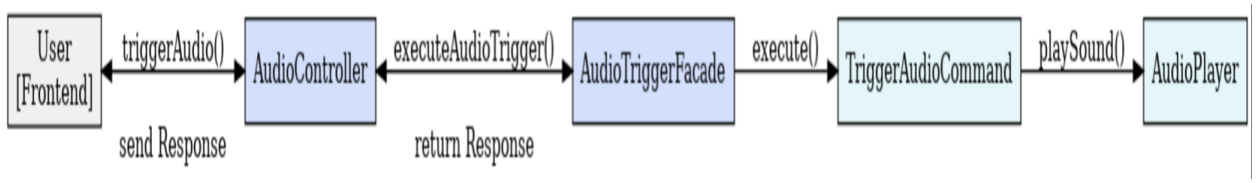


Figure 32: API_8 - Audio Trigger

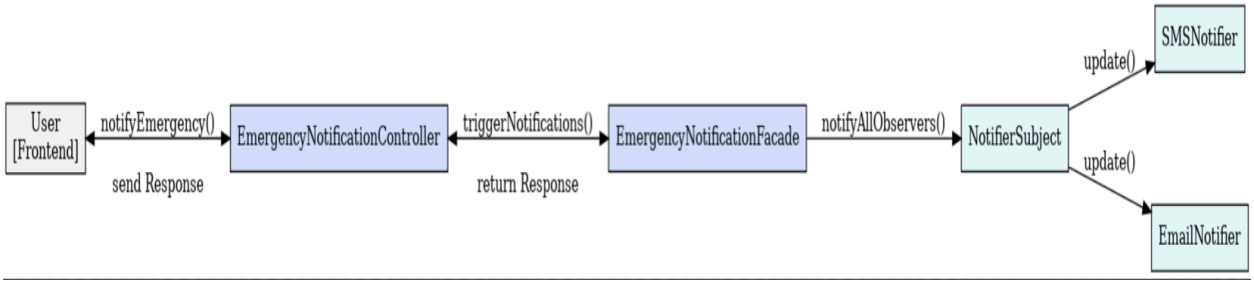


Figure 33: API_9 - Notify Emergency

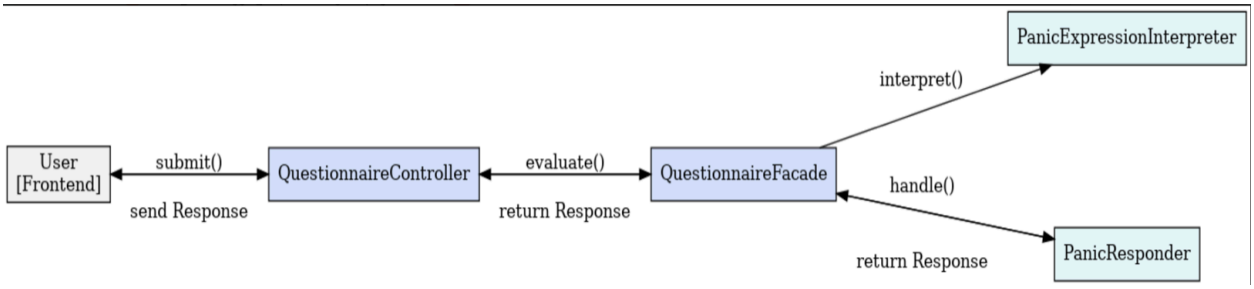


Figure 34: API_10 - Questionnaire Submit

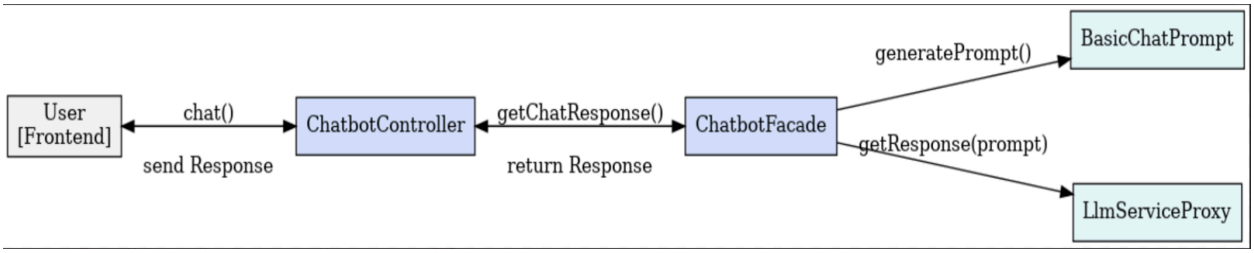


Figure 35: API_11 - Chatbot

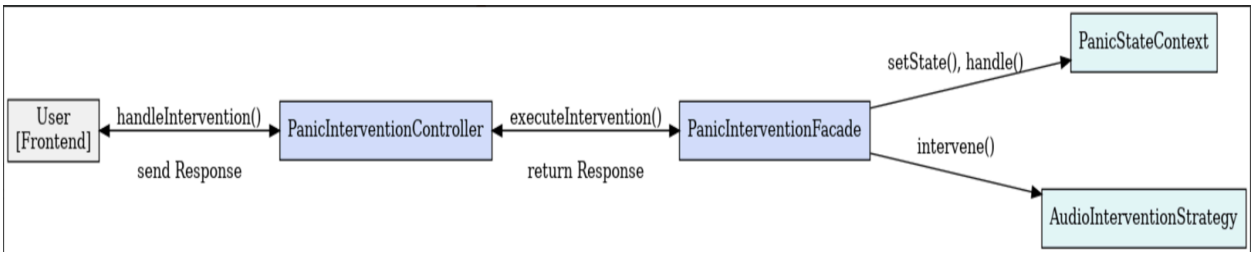
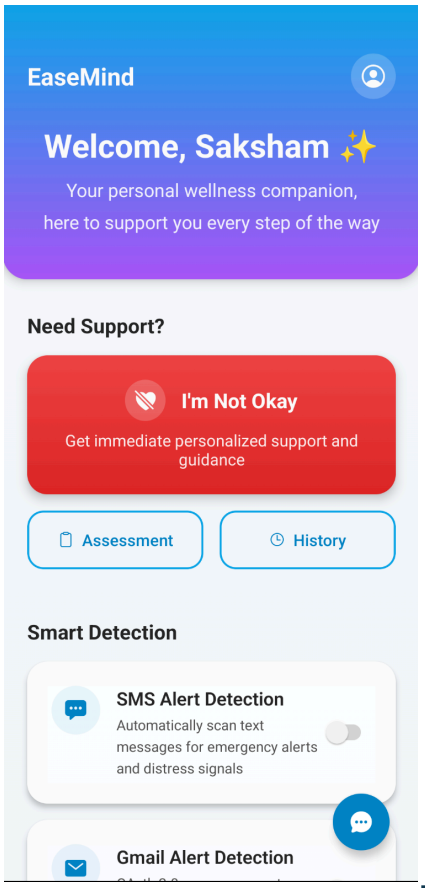
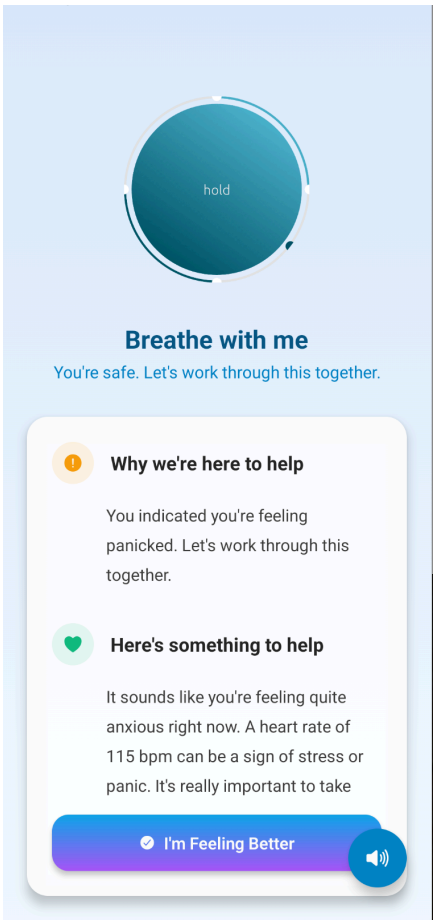
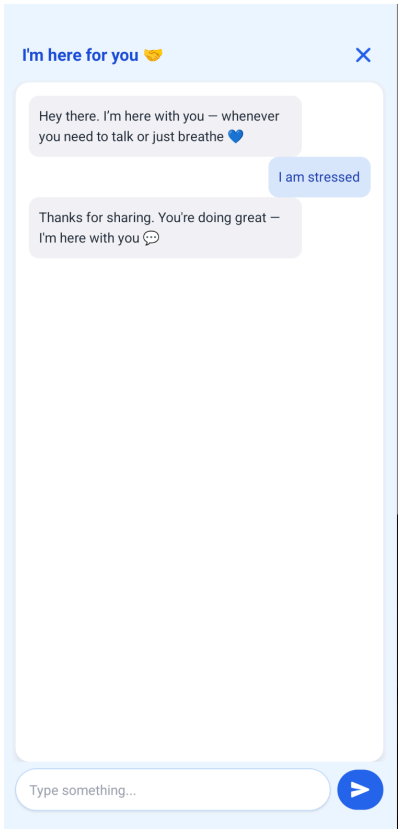


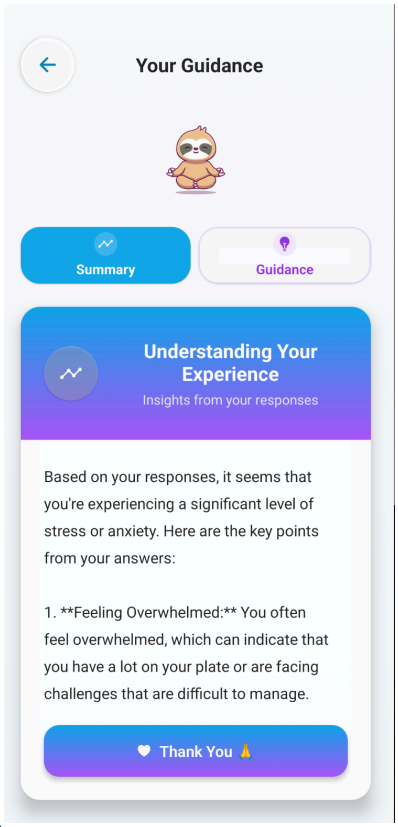
Figure 36: API_12 - Panic Intervention

Screenshots:









UI Documentation

Overview:

- **Chatbot Screen (React Native Gifted Chat)**
- **Mood Survey View (Styled Components + Lottie for animations)**
- **Notification via Adapter (react-native-push-notification)**
- **TTS Playback via Adapter (expo-speech)**

Challenges and Considerations:

1. **API Quotas/Billing on OpenRouter led to 402 errors; we built a mock fallback.**
2. **Model Token Limits required adding max_tokens to our request payload.**
3. **Bean Mapping Conflicts in Spring Boot when controllers overlapped—resolved by merging into a single PanicController.**

Future Work:

1. **Wearable Integration: HealthKit, Google Fit, Fitbit readers (Bridge pattern).**
2. **Voice Stress & Typing-Speed detectors via Visitor pattern.**
3. **Emergency Alerting: Twilio/SendGrid integration once wearables are live.**
4. **Personalization: Store user preferences to tune calming strategies over time.**

.