Elyx Multi-Agent Health Transformation System

Comprehensive Technical Documentation

AI-Powered Healthcare Coordination Platform

August 17, 2025

Contents

1 Executive Summary **3**
- 1.1 Key Features . . . 3

2 System Architecture **3**
- 2.1 Multi-Agent Framework . . . 3
- 2.2 Agent Roles and Responsibilities . . . 4

3 Program Structure **4**
- 3.1 32-Week Timeline . . . 4
- 3.2 Weekly Message Distribution . . . 4

4 Technical Implementation **4**
- 4.1 Base Agent Class Structure . . . 4
- 4.2 Message Generation Workflow . . . 5

5 Member Profile System **7**
- 5.1 Profile Structure . . . 7
- 5.2 Dynamic Profile Generation . . . 7

6 Communication System **7**
- 6.1 WhatsApp Integration . . . 7
- 6.2 Document Attachment System . . . 8

7 Temporal Management **8**
- 7.1 Chronological Timestamp Generation . . . 8
- 7.2 Travel Week Detection . . . 9

8 Conversation Dynamics **9**
- 8.1 Member-Initiated Conversations . . . 9
- 8.2 Complaint Scenario Integration . . . 9

9 Data Export and Analysis **9**
- 9.1 Conversation Logging . . . 9
- 9.2 Statistical Analysis . . . 10

10 Quality Assurance **10**
10.1 Message Validation 10
10.2 Fallback Mechanisms 10

11 Advanced Features **11**
11.1 Exercise Update Scheduling 11
11.2 Adaptive Agent Response Selection 11

12 Performance Optimization **11**
12.1 Generation Speed 11
12.2 Memory Management 12

13 Future Enhancements **12**
13.1 Potential Improvements 12
13.2 Scalability Considerations 12

14 Conclusion **12**

1 Executive Summary

The Elyx Multi-Agent Health Transformation System is a sophisticated AI-powered healthcare coordination platform designed to provide personalized health optimization services over a 32-week period. The system employs six specialized AI agents working collaboratively to deliver comprehensive health guidance through WhatsApp-based communication.

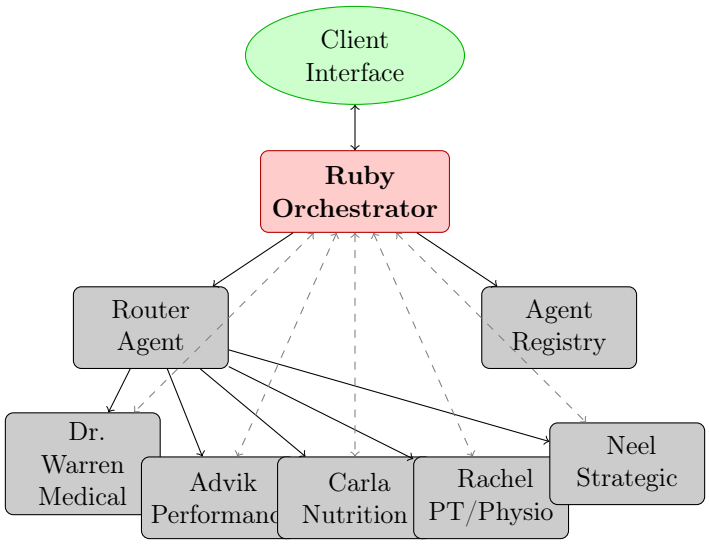
1.1 Key Features

- **Multi-Agent Architecture**: Six specialized AI agents with distinct expertise areas
- **32-Week Program**: Structured health transformation journey (January 15 - August 20, 2025)
- **WhatsApp Integration**: Mobile-first communication with document attachments
- **Travel-Aware**: Adapts to business travel patterns (1 week out of every 4)
- **Research-Driven**: Supports member-initiated research-based conversations
- **Realistic Simulation**: Generates 600+ chronologically ordered messages

2 System Architecture

2.1 Multi-Agent Framework

The system is built on a multi-agent architecture where each agent represents a specialized healthcare professional. The following diagram illustrates the agent hierarchy and interactions:



2.2 Agent Roles and Responsibilities

Agent	Role	Responsibilities
Ruby	Orchestrator & Concierge	Program coordination, scheduling, member support, travel logistics
Dr. Warren	Medical Strategist	Health assessments, lab interpretation, medical guidance, risk analysis
Advik	Performance Scientist	Data analysis, HRV monitoring, recovery insights, performance metrics
Carla	Nutritionist	Meal planning, dietary advice, supplements, Singapore-based nutrition
Rachel	PT/Physiotherapist	Exercise programming, form feedback, movement assessment, travel workouts
Neel	Concierge Lead	Strategic planning, long-term goals, program oversight, sustainability

Table 1: Agent Specializations

3 Program Structure

3.1 32-Week Timeline

The program is divided into eight distinct phases, each spanning 4 weeks:

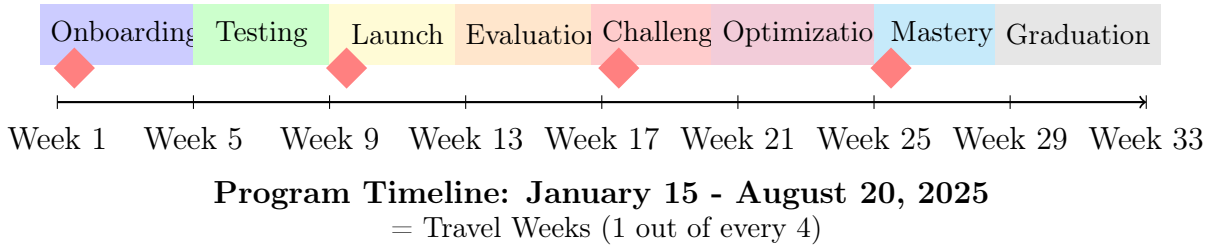


Figure 1: 32-Week Program Timeline with Travel Patterns

3.2 Weekly Message Distribution

Each week generates 18-22 messages following this pattern:

Figure 2: Weekly Message Distribution by Agent

4 Technical Implementation

4.1 Base Agent Class Structure

The system implements an abstract base class that all agents inherit from:

```python
class BaseAgent(ABC):
    def __init__(self, name: str, role: str, model_name: str,
       member_profile: Dict):
        self.name = name
        self.role = role
        self.member_profile = member_profile
        self.llm = ChatGoogleGenerativeAI(model=model_name,
           temperature=self.get_temperature())
        self.conversation_history = []

    @abstractmethod
    def get_system_prompt(self) -> str:
        pass

    @abstractmethod
    def get_temperature(self) -> float:
        pass

    def generate_week_messages(self, week_number: int, week_focus:
        str,
                               message_count: int,
                                  conversation_type: str) -> List[
                                  Tuple[str, str]]:
        # Implementation for weekly message generation
        pass

    def process_message(self, message: str, context: str) -> str:
        # Individual message processing
        pass
```

Listing 1: Base Agent Implementation

4.2 Message Generation Workflow

The following flowchart illustrates the message generation process:

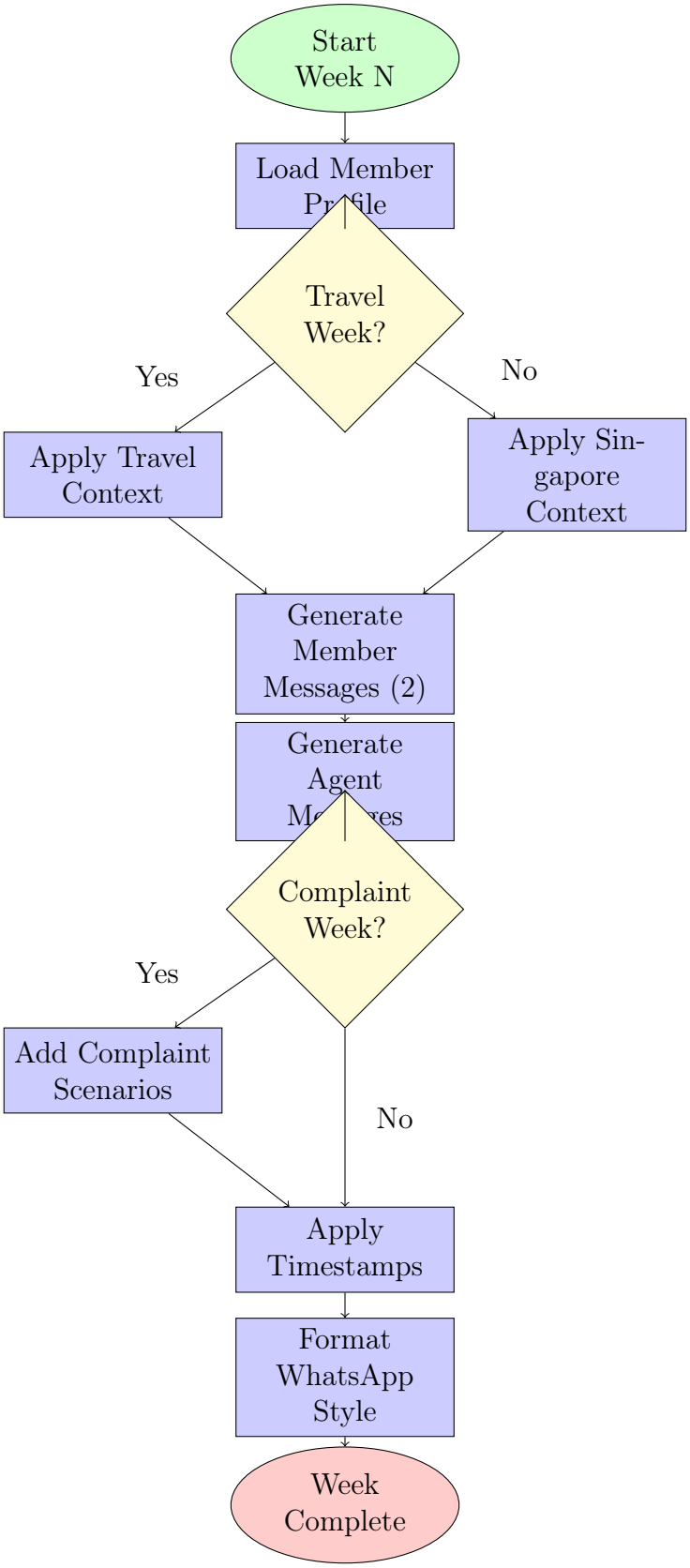


Figure 3: Message Generation Workflow

5 Member Profile System

5.1 Profile Structure

Each member profile contains comprehensive information organized into categories:

Category	Components
Snapshot	Name, DOB, Age, Gender, Residence, Travel hubs, Occupation, Assistant
Outcomes	Goals (3 specific objectives), Motivation, Key metrics
Behavioral	Personality traits, Motivation stage, Support network, Mental health
Tech Stack	Wearables (Apple Watch/Garmin), Apps, Data sharing preferences, Reporting
Communication	WhatsApp preference, Response time expectations, Detail depth, Language
Scheduling	Availability, Travel calendar, Appointment mix, Transport arrangements

Table 2: Member Profile Components

5.2 Dynamic Profile Generation

The system can generate realistic member profiles using predefined constraints:

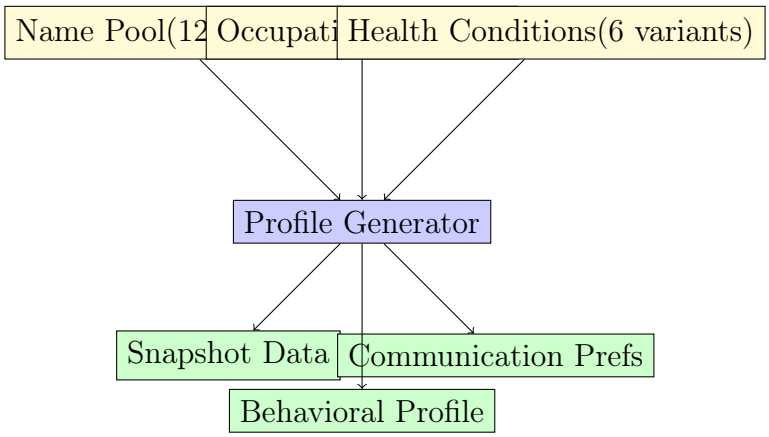


Figure 4: Dynamic Profile Generation Process

6 Communication System

6.1 WhatsApp Integration

The system exclusively uses WhatsApp-style communication with specific formatting:

```
def format_message(sender: str, message: str, timestamp: str) ->
    str:
    if sender in role_titles:
        role = role_titles[sender]
        return f"{timestamp} {sender} (Elyx {role}): {message}"
```

```
    else:
        return f"{timestamp} {sender}: {message}"

# Example output:
# [01/15/25, 9:00 AM] Ruby (Elyx Orchestrator & Concierge):
# Good morning Rohan! Welcome to your health transformation
   journey...
```

Listing 2: WhatsApp Message Formatting

6.2 Document Attachment System

The system references document attachments in WhatsApp format:

- `diet_plan_week5.txt` - Nutritional guidance
- `workout_routine_week10.txt` - Exercise programs
- `blood_report_analysis.txt` - Lab results interpretation
- `sleep_optimization.txt` - Recovery recommendations
- `travel_protocol.txt` - Business travel adaptations

7 Temporal Management

7.1 Chronological Timestamp Generation

Messages within each week are chronologically ordered using realistic business hours:

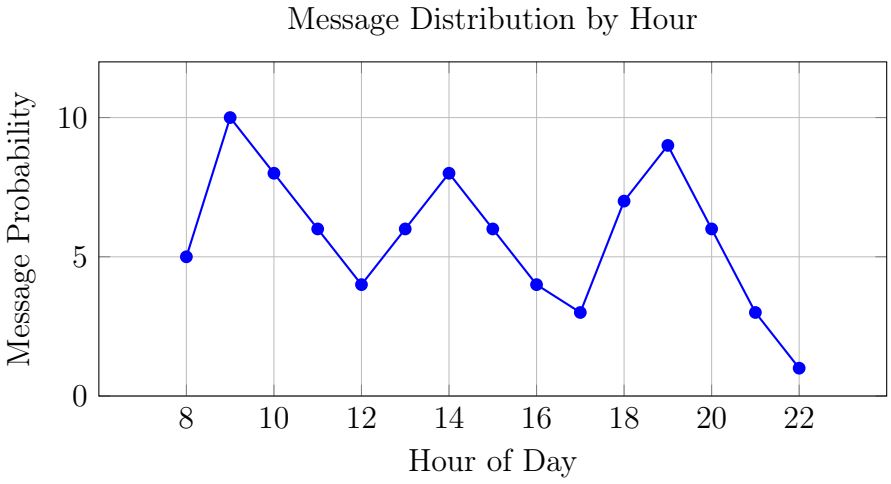


Figure 5: Hourly Message Distribution Pattern

7.2 Travel Week Detection

The system automatically detects travel weeks using a modular pattern:

$$\text{Travel Week} = \begin{cases} \text{True} & \text{if } (week\_number - 1) \bmod 4 = 0 \\ \text{False} & \text{otherwise} \end{cases} \tag{1}$$

This results in travel weeks: 1, 5, 9, 13, 17, 21, 25, 29, 33...

8 Conversation Dynamics

8.1 Member-Initiated Conversations

Each week includes exactly 2 member-initiated conversations based on health research:

Research Topic	Example Question
Intermittent Fasting	"I read a study about 16:8 fasting improving insulin sensitivity. How does this apply to my travel schedule?"
Sleep Optimization	"Found research on HRV and sleep quality correlation. Can we integrate this with my Garmin data?"
Travel Recovery	"Article about jet lag supplements - do any of these conflict with my current protocol?"
Mediterranean Diet	"Study shows Mediterranean diet reduces inflammation. How do I adapt this for Singapore dining?"

Table 3: Member Research Question Examples

8.2 Complaint Scenario Integration

The system includes realistic complaint scenarios at specific weeks:

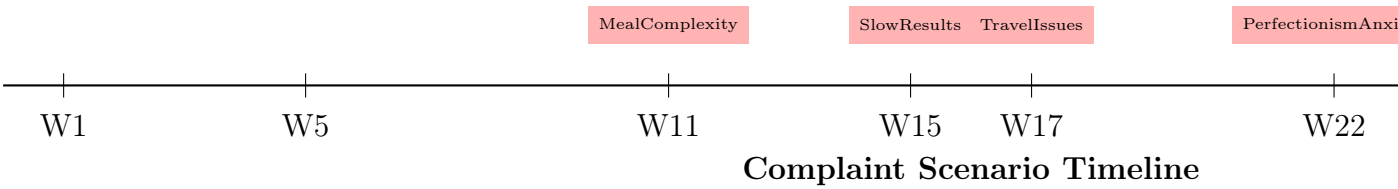


Figure 6: Complaint Scenarios Distribution

9 Data Export and Analysis

9.1 Conversation Logging

The system maintains comprehensive logs of all interactions:

```
self.conversation_log.append({
    'week': week_number,
    'sender': sender,
    'message': message,
```

```
    'timestamp': timestamp,
    'formatted': formatted_message,
    'travel_week': travel_week
})
```

Listing 3: Conversation Log Structure

9.2 Statistical Analysis

The system generates detailed statistics for each run:

Metric	Typical Values
Total Messages	600-700 (for 32 weeks)
Average per Week	18-22 messages
Travel Weeks	8 out of 32 weeks (25%)
Research Questions	64 total (2 per week)
Complaint Scenarios	5 specific weeks
Agent Distribution	Ruby 25%, Others 15% each, Member 20%

Table 4: System Performance Metrics

10 Quality Assurance

10.1 Message Validation

The system implements multiple validation layers:

1. **Sender Verification**: Ensures only valid agents and member names are used
2. **Message Count Control**: Maintains 18-22 messages per week
3. **Timestamp Ordering**: Chronological ordering within each week
4. **Context Consistency**: Travel/Singapore context applied correctly
5. **Role Appropriateness**: Messages match agent expertise areas

10.2 Fallback Mechanisms

When AI generation fails, the system provides intelligent fallbacks:

```
# Fallback messages when AI generation fails
fallback_messages = []
for i in range(message_count):
    if i % 2 == 0:
        if travel_week:
            fallback_messages.append((self.name,
                f"Hope your business trip is going well! Week {
                    week_number} check-in."))
        else:
            fallback_messages.append((self.name,
```

```
                f"Week {week_number} check-in from {self.name}."))
    else:
        fallback_messages.append((member_name,
            f"Thanks {self.name}, managing well."))
```

Listing 4: Fallback Message Generation

11 Advanced Features

11.1 Exercise Update Scheduling

The system automatically schedules exercise updates every 2 weeks:

$$\text{Exercise Update Week} = \begin{cases} \text{True} & \text{if } week\_number \bmod 2 = 0 \\ \text{False} & \text{otherwise} \end{cases} \tag{2}$$

11.2 Adaptive Agent Response Selection

The system intelligently selects responding agents based on message content:

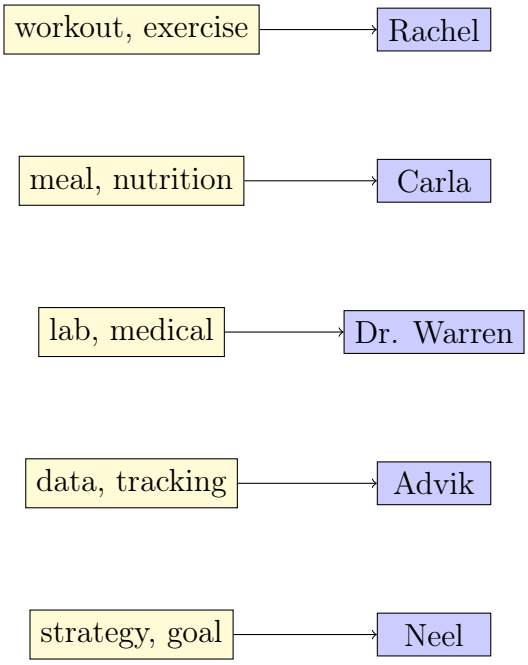


Figure 7: Keyword-Based Agent Selection

12 Performance Optimization

12.1 Generation Speed

The system includes several optimizations for efficient generation:

- **Batch Processing**: Generates multiple messages per agent call

- **Caching**: Reuses member profile data across agents
- **Parallel Processing**: Potential for concurrent agent execution
- **Smart Fallbacks**: Rapid recovery from generation failures

12.2 Memory Management

Each agent maintains conversation history that can be reset:

```
def reset_all_histories(self) -> None:
    for agent in self.agents.values():
        agent.reset_history()
```

Listing 5: Memory Management

13 Future Enhancements

13.1 Potential Improvements

1. **Real WhatsApp Integration**: Connect to actual WhatsApp Business API
2. **Machine Learning**: Learn from member preferences over time
3. **Voice Messages**: Support for audio content generation
4. **Multi-Language**: Support for multiple languages beyond English
5. **Real-Time Scheduling**: Integration with calendar systems
6. **Biometric Integration**: Connect to actual wearable devices

13.2 Scalability Considerations

- **Database Integration**: Store conversations persistently
- **API Optimization**: Reduce LLM API calls through intelligent caching
- **Load Balancing**: Distribute agent processing across multiple instances
- **Monitoring**: Add comprehensive logging and error tracking

14 Conclusion

The Elyx Multi-Agent Health Transformation System represents a sophisticated approach to personalized healthcare coordination. By leveraging specialized AI agents, realistic conversation simulation, and comprehensive member profiling, the system creates engaging and authentic health transformation journeys.

The system's strength lies in its:

- **Realistic Simulation**: Generates believable conversations with appropriate timing

- **Specialized Expertise**: Each agent provides domain-specific guidance
- **Adaptive Behavior**: Responds to member needs and travel patterns
- **Comprehensive Coverage**: Addresses all aspects of health transformation
- **Quality Assurance**: Multiple validation layers ensure consistency

This documentation provides a complete technical reference for understanding, implementing, and extending the Elyx Multi-Agent Health Transformation System.