DOGEv2

# Persona System Design

## Theoretical Foundations

### Psychological Models

#### 1. Five Factor Model (FFM/OCEAN)

The system's core personality framework is built on the Five Factor Model (Costa & McCrae, 1992), which provides empirically validated dimensions of personality:

- **Openness to Experience**: Influences curiosity, creativity, and abstract thinking
- **Conscientiousness**: Determines organization, responsibility, and goal-oriented behavior
- **Extraversion**: Shapes social interaction patterns and energy expression
- **Agreeableness**: Affects cooperation, empathy, and conflict resolution
- **Neuroticism**: Modulates emotional stability and stress response

#### 2. Attachment Theory

Building on Bowlby's (1969) and Ainsworth's (1978) work, the persona system implements:

- Secure, anxious, and avoidant attachment styles
- Dynamic relationship formation patterns
- Trust threshold mechanisms
- Emotional regulation strategies

#### 3. Emotional Intelligence Framework

Based on Salovey and Mayer's (1990) model, enhanced by Goleman's (1995) work:

- Self-awareness components
- Emotional regulation mechanisms
- Social skill implementation
- Empathy processing systems

# Cognitive Science Integration

## 1. Memory Systems

Implementing Atkinson-Shiffrin's (1968) memory model with modern updates:

- **Episodic Memory**: Core experiences and their emotional signatures
- **Semantic Memory**: Knowledge and belief networks
- **Procedural Memory**: Communication patterns and behavioral scripts

## 2. Belief Systems

Based on cognitive consistency theories (Festinger, 1957):

- Belief network interconnections
- Cognitive dissonance handling
- Belief strength modulation
- Value hierarchy implementation

# Sociolinguistic Framework

## 1. Communication Accommodation Theory (Giles, 1973)

Implements dynamic communication style adaptation:

- Context-dependent language patterns
- Social role influences
- Cultural code-switching

- Status-based modifications

### 2. Discourse Analysis Integration

Based on conversation analysis principles (Sacks, Schegloff, & Jefferson, 1974):

- Turn-taking mechanisms
- Topic transition patterns
- Repair strategies
- Contextual relevance maintenance

# Implementation Architecture

## System Design Rationale

### 1. Cognitive Architecture Foundations

The system's architecture draws from several key cognitive theories:

1. **ACT-R Theory (Anderson, 1996)**
   - Distinction between declarative and procedural knowledge
   - Production rule system for behavior generation
   - Activation-based memory retrieval
   - Learning through pattern strengthening
2. **Parallel-Distributed Processing (Rumelhart & McClelland, 1986)**
   - Network-based knowledge representation
   - Weighted connections between concepts
   - Dynamic pattern activation
   - Emergent behavioral properties
3. **Working Memory Model (Baddeley, 2000)**
   - Multiple component memory system
   - Phonological loop for language processing
   - Episodic buffer for multimodal integration

- Central executive for attention control

## 2. Hybrid Storage Implementation

1. **PostgreSQL (Declarative Knowledge)**
   Implements Anderson's declarative memory concepts:
   - Explicit fact storage (biographical data)
   - Attribute-value pairs (personality traits)
   - Temporal markers (experience timestamps)
   - Activation history (usage patterns)
     Performance considerations:
   - JSONB for flexible schema evolution
   - GiST indexing for complex queries
   - Materialized views for frequent patterns
2. **Knowledge Graph (Procedural & Episodic Knowledge)**
   Based on spreading activation theories (Collins & Loftus, 1975):
   - Dynamic relationship networks
   - Experience-behavior mappings
   - Belief propagation patterns
   - Interactive learning systems
     Implementation features:
   - Graph-based pattern matching
   - Weighted relationship edges
   - Temporal relationship evolution
   - Contextual activation paths

## 3. Knowledge Graph Architecture

1. **Theoretical Foundations**
   Drawing from cognitive network theories:
   - Spreading Activation Theory (Collins & Loftus, 1975)
   - Semantic Networks (Quillian, 1967)
   - Connectionist Models (Rumelhart & McClelland, 1986)

- Schema Theory (Bartlett, 1932)

2. **Implementation Structure**

```
interface KnowledgeNode {
  type: 'concept' | 'experience' | 'pattern' | 'belief';
  weight: number;  // Activation strength
  connections: Connection[];
  temporalMarkers: {
    created: Date;
    lastActivated: Date;
    activationCount: number;
  };
}

interface Connection {
  target: KnowledgeNode;
  type: 'influences' | 'causes' | 'relates_to' | 'contradicts';
  strength: number;  // Connection weight
  context: string[];  // Activation contexts
}
```

3. **Dynamic Processing**
   Based on neural network principles (Hopfield, 1982):
   - Activation spreading through weighted connections
   - Context-sensitive pattern activation
   - Temporal decay of connection strengths
   - Hebbian learning for pattern reinforcement

4. **Interaction Patterns**
   Implementing social cognition models (Fiske & Taylor, 1991):
   - Script-based behavior generation
   - Role-based interaction patterns
   - Cultural schema activation
   - Social norm compliance

## 4. Empirical Validation

The system's effectiveness is supported by research in:

1. **Personality Psychology**
   - Longitudinal studies of trait stability (Roberts & DelVecchio, 2000)
   - Cross-cultural personality research (McCrae & Terracciano, 2005)
   - Behavioral prediction models (Funder, 2001)
   - Dynamic personality processes (Fleeson, 2001)
2. **Cognitive Psychology**
   - Memory systems research (Squire & Zola, 1996)
   - Decision-making studies (Kahneman, 2011)
   - Learning pattern analysis (Kolb, 1984)
   - Schema activation studies (Brewer & Treyens, 1981)
3. **Sociolinguistics**
   - Discourse analysis findings (Tannen, 1993)
   - Conversation structure studies (Heritage, 2008)
   - Pragmatic adaptation research (Kasper, 1997)
   - Sociolinguistic variation (Labov, 2001)
4. **Network Science**
   - Small-world network properties (Watts & Strogatz, 1998)
   - Scale-free networks in cognition (Barabási & Albert, 1999)
   - Dynamic network evolution (Newman, 2003)
   - Information propagation patterns (Granovetter, 1973)

# Core Components

# Enhanced Persona Template

```
interface Persona {
  // Basic Information
  id: string;
  name: string;
  version: string;
```

```
  // Voice Characteristics
  voice: {
    baseProfile: string;
    pitch: number;
    speed: number;
    clarity: number;
    accent: string;
    dialectPatterns: string[];
    emotionalModulation: {
      happiness: number;
      anger: number;
      sadness: number;
      fear: number;
    };
  };

  // Personality Framework
  personality: {
    // Big Five (OCEAN)
    traits: {
      openness: number;
      conscientiousness: number;
      extraversion: number;
      agreeableness: number;
      neuroticism: number;
    };

    // Attachment Style
    attachment: {
      secure: number;
      anxious: number;
      avoidant: number;
    };

    // Cultural Intelligence
    culturalQuotient: {
      awareness: number;
      knowledge: number;
      motivation: number;
```

```
      behavior: number;
    };

    // Emotional Intelligence
    emotionalQuotient: {
      selfAwareness: number;
      selfRegulation: number;
      motivation: number;
      empathy: number;
      socialSkills: number;
    };
  };

  // Core Memory System
  coreMemories: {
    experiences: Array<{
      type: 'achievement' | 'trauma' | 'relationship' |
'learning';
      impact: number; // -1 to 1
      age: number;
      description: string;
      relatedTopics: string[];
      emotionalSignature: EmotionalState;
    }>;

    beliefs: Array<{
      category: 'moral' | 'political' | 'religious' |
'scientific' | 'personal';
      value: string;
      strength: number; // 0 to 1
      flexibility: number; // 0 to 1
      sources: string[];
      connectedBeliefs: string[];
    }>;

    relationships: Array<{
      type: string;
      impact: number;
      lessons: string[];
```

```
    trustThreshold: number;
  }>;
};

// Knowledge Framework
knowledge: {
  expertise: Array<{
    domain: string;
    level: number;
    confidence: number;
    applications: string[];
  }>;

  experiences: Array<{
    context: string;
    significance: number;
    learnings: string[];
  }>;

  biases: Array<{
    type: string;
    strength: number;
    triggers: string[];
  }>;
};

// Communication Framework
communication: {
  // Context-dependent styles
  styles: Map<string, {
    formality: number;
    directness: number;
    emotionalExpression: number;
    vocabularyLevel: number;
    humorIncidence: number;
  }>;

  patterns: {
    catchphrases: string[];
```

```
      fillerWords: string[];
      sentenceStructures: string[];
      topicTransitions: Array<{
        from: string;
        to: string;
        method: string;
      }>;
    };

    // Conflict handling
    conflictStyle: {
      defaultApproach: 'compete' | 'collaborate' | 'compromise'
| 'avoid' | 'accommodate';
      escalationThreshold: number;
      deescalationTechniques: string[];
      triggerTopics: string[];
    };
  };

  // Growth & Learning
  development: {
    currentGoals: string[];
    learningRate: number;
    adaptability: number;
    recentExperiences: Array<{
      type: string;
      impact: number;
      dateOccurred: Date;
    }>;
  };
}
```

# Emotional Modeling

## Emotional States

```
interface EmotionalState {
  primary: Emotion;
```

```
  intensity: number;
  duration: number;
  decay: number;

  modifiers: {
    topic: string;
    multiplier: number;
  }[];
}

type Emotion =
  | 'neutral'
  | 'happy'
  | 'angry'
  | 'sad'
  | 'excited'
  | 'anxious'
  | 'confused'
  | 'defensive';
```

# Interaction Patterns

## Conversation Dynamics

```
interface ConversationBehavior {
  // Turn Taking
  interruptionThreshold: number;
  responseDelay: number;
  turnDuration: number;

  // Topic Management
  topicPreferences: string[];
  topicAvoidance: string[];
  tangentProbability: number;

  // Interaction Style
  agreementTendency: number;
  elaborationLevel: number;
```

```
  questionFrequency: number;
}
```

# Implementation Guide

## 1. Hybrid Persona Creation

```
class EnhancedPersonaBuilder {
  private persona: Persona;
  private knowledgeBase: KnowledgeGraph;
  private storage: PostgresStorage;

  constructor() {
    this.persona = this.initializeTemplate();
    this.knowledgeBase = new KnowledgeGraph(mcp.memory);
    this.storage = new PostgresStorage(mcp.postgres);
  }

  public async setVoiceProfile(profile: VoiceProfile):
Promise<this> {
    // Query knowledge base for voice pattern insights
    const patterns = await
this.knowledgeBase.query('voice_patterns', {
      accent: profile.accent,
      dialect: profile.dialectPatterns
    });

    // Enhance voice profile with learned patterns
    this.persona.voice = {
      ...profile,
      ...patterns.recommendations
    };
    return this;
  }

  public async setPersonality(traits: PersonalityTraits):
Promise<this> {
    // Get personality insights from knowledge graph
```

```
    const insights = await
this.knowledgeBase.query('personality_patterns', {
      traits: traits
    });

    // Merge traits with knowledge-based insights
    this.persona.personality = {
      ...traits,
      ...insights.behavioral_patterns
    };
    return this;
  }

  public async build(): Promise<Persona> {
    // Validate against both storage systems
    await this.validateWithKnowledge();
    await this.validateWithStorage();

    // Store in PostgreSQL
    const id = await this.storage.create(this.persona);

    // Add to knowledge graph
    await this.knowledgeBase.addEntity({
      name: `persona_${id}`,
      type: 'Persona',
      traits: this.persona.personality,
      patterns: this.persona.speech
    });

    return this.persona;
  }

  private async validateWithKnowledge(): Promise<void> {
    const validation = await
this.knowledgeBase.validate('persona_rules', this.persona);
    if (!validation.valid) {
      throw new Error(`Knowledge validation failed:
${validation.reasons.join(', ')}`);
    }
```

```
  }
}
```

## 2. Emotional Processing

```
class EmotionalProcessor {
  private currentState: EmotionalState;

  public processStimulus(
    stimulus: ConversationEvent
  ): EmotionalResponse {
    const impact = this.calculateEmotionalImpact(stimulus);
    return this.updateEmotionalState(impact);
  }

  private calculateEmotionalImpact(
    stimulus: ConversationEvent
  ): EmotionalImpact {
    // Implementation details
  }
}
```

## 3. Interaction Management

```
class InteractionManager {
  private personas: Map<string, Persona>;
  private currentContext: ConversationContext;

  public initializeInteraction(
    participants: string[]
  ): void {
    this.loadPersonas(participants);
    this.establishContext();
  }

  public generateResponse(
    stimulus: ConversationEvent
```

```
  ): Response {
    const emotionalResponse = this.processEmotions(stimulus);
    return this.createResponse(emotionalResponse);
  }
}
```

# Voice Profile Integration

## Voice Characteristics

```
interface VoiceProfile {
  baseVoice: string;
  modifications: {
    pitch: number;
    speed: number;
    emotionalModulation: boolean;
  };
  effects: {
    reverb: number;
    compression: number;
    eq: FrequencyResponse[];
  };
}
```

# Storage and Retrieval

## Database Schema

```
-- Core persona table with JSONB for flexible schema evolution
CREATE TABLE personas (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  version VARCHAR(50) NOT NULL,
  persona_data JSONB NOT NULL,
  metadata JSONB NOT NULL,
  created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
```

```
  usage_count INTEGER DEFAULT 0,
  last_used TIMESTAMPTZ
);

-- Efficient indexing for common queries
CREATE INDEX idx_personas_name ON personas (name);
CREATE INDEX idx_personas_version ON personas (version);
CREATE INDEX idx_personas_metadata ON personas USING GIN
(metadata);
CREATE INDEX idx_personas_persona_data ON personas USING GIN
(persona_data);

-- Trigger to automatically update updated_at
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$ language 'plpgsql';

CREATE TRIGGER update_personas_updated_at
    BEFORE UPDATE ON personas
    FOR EACH ROW
    EXECUTE FUNCTION update_updated_at_column();
```

## MCP Integration

```
// Using the PostgreSQL MCP tool for persona operations
interface PersonaStorage {
  async create(persona: Persona): Promise<number> {
    const result = await mcp.postgres.query(`
      INSERT INTO personas (name, version, persona_data,
metadata)
      VALUES ($1, $2, $3, $4)
      RETURNING id
    `, [persona.name, persona.version, persona, { created: new
Date() }]);
```

```
    return result.rows[0].id;
  }

  async update(id: number, persona: Persona): Promise<void> {
    await mcp.postgres.query(`
      UPDATE personas
      SET persona_data = $1, metadata = metadata || $2
      WHERE id = $3
    `, [persona, { modified: new Date() }, id]);
  }

  async get(id: number): Promise<Persona> {
    const result = await mcp.postgres.query(`
      SELECT persona_data FROM personas WHERE id = $1
    `, [id]);
    return result.rows[0].persona_data;
  }
}
```

# Testing Framework

## Unit Tests

```
describe('Persona System', () => {
  it('should create valid personas', () => {
    const builder = new PersonaBuilder();
    const persona = builder
      .setVoiceProfile(defaultVoice)
      .setPersonality(defaultTraits)
      .build();

    expect(persona).toBeDefined();
    expect(persona.voice).toBeDefined();
  });
});
```

# Related Documents

- MCP Architecture Overview
- Script Generation Engine
- Audio Production Pipeline

# Implementation Checklist

- [ ] Set up persona database
- [ ] Implement persona builder
- [ ] Create emotion processor
- [ ] Develop interaction manager
- [ ] Integrate voice profiles
- [ ] Implement storage system
- [ ] Deploy testing framework