S.C.O.U.T. v3: Backend API Architecture & Documentation

System Architecture Overview

The S.C.O.U.T. v3 backend is built as a microservices architecture using FastAPI with Python 3.11, featuring:

- Event-driven architecture with message queues
- Microservices for specialized AI processing
- Real-time WebSocket connections
- Comprehensive caching and optimization
- Auto-scaling containerized deployment

Technology Stack

Core Framework:

- FastAPI 0.104.0 (Python 3.11+)
- Pydantic 2.4.0 for data validation
- SQLAlchemy 2.0 for database operations
- Alembic for database migrations

AI & Machine Learning:

- OpenAI API (GPT-4 Turbo, Whisper, DALL-E)
- Azure Cognitive Services
- Custom ML models with TensorFlow 2.13
- Hugging Face Transformers
- spaCy for NLP processing

Infrastructure:

- PostgreSQL 15 with pgvector extension
- Redis 7.0 for caching and session management
- Celery for background task processing
- Docker + Kubernetes for containerization
- Azure Cloud Services (AKS, Blob Storage, Service Bus)

Real-time & Communication:

- WebSocket connections via FastAPI WebSocket

- Azure SignalR for scaling real-time connections
- Azure Service Bus for message queuing
- SendGrid for email services

Database Schema

Core Tables

```
-- Candidates table
CREATE TABLE candidates (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    name VARCHAR(255) NOT NULL,
    phone VARCHAR(20),
    position VARCHAR(255) NOT NULL,
    experience_level VARCHAR(50),
    source VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Assessments table
CREATE TABLE assessments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    candidate_id UUID REFERENCES candidates(id),
    template_id UUID REFERENCES assessment_templates(id),
    status VARCHAR(50) DEFAULT 'invited',
    invitation_sent_at TIMESTAMP,
    started_at TIMESTAMP,
    completed_at TIMESTAMP,
    expires_at TIMESTAMP,
    overall_score INTEGER,
    recommendation VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Assessment responses table
CREATE TABLE assessment_responses (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    assessment_id UUID REFERENCES assessments(id),
    question_id VARCHAR(255) NOT NULL,
    response_text TEXT,
    response_metadata JSONB,
    ai_analysis JSONB,
    score INTEGER,
    time_taken INTEGER,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Multimodal analysis data
CREATE TABLE multimodal_analysis (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    assessment_id UUID REFERENCES assessments(id),
    analysis_type VARCHAR(50), -- 'voice', 'facial', 'behavioral'
    raw_data JSONB,
    processed_features JSONB,
    confidence_score FLOAT,
    insights JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Predictive models table
CREATE TABLE predictive_scores (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    candidate_id UUID REFERENCES candidates(id),
    model_type VARCHAR(100),
    prediction_data JSONB,
    confidence_score FLOAT,
    model_version VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

API Endpoints Specification

1. AUTHENTICATION & AUTHORIZATION

1.1 Admin Authentication

POST /api/v3/auth/login

```
# Admin login with MFA support
@router.post("/auth/login", response_model=TokenResponse)
async def admin_login(credentials: AdminLoginRequest):
    """
    Authenticate admin user with multi-factor authentication

    Request Body:
    {
        "email": "admin@studai.one",
        "password": "secure_password",
        "mfa_token": "123456"  # Optional for first login
    }

    Response:
    {
        "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",
        "refresh_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",
        "token_type": "bearer",
        "expires_in": 3600,
        "user": {
            "id": "usr_001",
            "name": "Priya Sharma",
            "role": "HR_MANAGER",
            "permissions": ["view_assessments", "create_invitations"]
        }
    }
    """
```

POST /api/v3/auth/refresh

```
# Token refresh endpoint
@router.post("/auth/refresh", response_model=TokenResponse)
async def refresh_token(refresh_request: RefreshTokenRequest):
    """
    Refresh access token using refresh token

    Request Body:
    {
        "refresh_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..."
    }
    """
```

1.2 Candidate Authentication

GET /api/v3/candidate/verify/{token}

```
# Candidate email verification and authentication
@router.get("/candidate/verify/{token}", response_model=CandidateSession)
async def verify_candidate_token(token: str):
    """
    Verify candidate invitation token and create session

    Response:
    {
        "session_id": "sess_cnd_001247",
        "candidate": {
            "id": "CND_2025_001247",
            "name": "Arjun Krishnamurthy",
            "email": "arjun.k@example.com",
            "position": "Senior AI Engineer"
        },
        "assessment": {
            "id": "ASS_AI_ENG_20250915_001",
            "template": "AI_ENGINEER_SENIOR_V3",
            "estimated_duration": 30,
            "expires_at": "2025-09-20T18:00:00Z"
        }
    }
    """
```

2. CANDIDATE MANAGEMENT

2.1 Candidate Operations

POST /api/v3/admin/candidates

```
# Create new candidate and send invitation
@router.post("/admin/candidates", response_model=CandidateResponse)
async def create_candidate(candidate_data: CreateCandidateRequest):
    """
    Create candidate profile and send assessment invitation

    Request Body:
    {
        "name": "Kavya Reddy",
        "email": "kavya.reddy@email.com",
        "phone": "+91-9876543210",
        "position": "UX Designer",
        "experience_level": "mid_level",
        "source": "linkedin",
        "assessment_template": "UX_DESIGNER_STANDARD",
        "custom_message": "We're excited to learn more about your design expertise!",
        "send_invitation": true,
        "validity_days": 7
    }

    Response:
    {
        "candidate_id": "CND_2025_001251",
        "invitation_sent": true,
        "assessment_link": "https://scout.studai.one/verify/eyJ0eXAi...",
        "expires_at": "2025-09-22T18:00:00Z"
    }
    """
```

GET /api/v3/admin/candidates

```
# List candidates with filtering and pagination
@router.get("/admin/candidates", response_model=CandidateListResponse)
async def list_candidates(
    page: int = 1,
    size: int = 20,
    status: Optional[str] = None,
    position: Optional[str] = None,
    score_min: Optional[int] = None,
    date_from: Optional[datetime] = None,
    search: Optional[str] = None
):
    """
    Retrieve paginated list of candidates with filters

    Query Parameters:
    - page: Page number (default: 1)
    - size: Items per page (default: 20, max: 100)
    - status: Filter by assessment status
    - position: Filter by job position
    - score_min: Minimum overall score
    - date_from: Filter by creation date
    - search: Search in name or email

    Response:
    {
        "candidates": [
            {
                "id": "CND_2025_001248",
                "name": "Ananya Gupta",
                "email": "ananya.g@email.com",
                "position": "Data Scientist",
                "status": "completed",
                "overall_score": 91,
                "recommendation": "highly_recommended",
                "submitted_at": "2025-09-14T16:45:00Z",
                "tags": ["Machine Learning", "Python", "High Potential"]
            }
        ],
        "pagination": {
            "page": 1,
            "size": 20,
            "total": 156,
            "pages": 8
        }
    }
    """
```

2.2 Bulk Operations

POST /api/v3/admin/candidates/bulk

```
# Bulk candidate invitation
@router.post("/admin/candidates/bulk", response_model=BulkOperationResponse)
async def bulk_create_candidates(bulk_data: BulkCandidateRequest):
    """
    Create multiple candidates and send invitations

    Request Body:
    {
        "candidates": [
            {
                "name": "Raj Patel",
                "email": "raj.patel@email.com",
                "position": "Product Manager",
                "experience_level": "senior"
            },
            {
                "name": "Meera Singh",
                "email": "meera.singh@email.com",
                "position": "Data Scientist",
                "experience_level": "mid_level"
            }
        ],
        "default_template": "GENERAL_ASSESSMENT_V3",
        "validity_days": 7,
        "send_invitations": true
    }

    Response:
    {
        "processed": 2,
        "successful": 2,
        "failed": 0,
        "results": [
            {
                "candidate_id": "CND_2025_001252",
                "email": "raj.patel@email.com",
                "status": "invited",
                "invitation_sent": true
            }
        ],
        "errors": []
    }
    """
```

3. ASSESSMENT ENGINE

3.1 Assessment Session Management

POST /api/v3/assessment/start

```
# Start assessment session
@router.post("/assessment/start", response_model=AssessmentSessionResponse)
async def start_assessment(session_data: StartAssessmentRequest):
    """
    Initialize assessment session for candidate

    Request Body:
    {
        "candidate_token": "eyJ0eXAiOiJKV1QiLCJ...",
        "browser_info": {
            "user_agent": "Mozilla/5.0...",
            "screen_resolution": "1920x1080",
            "timezone": "Asia/Kolkata"
        }
    }

    Response:
    {
        "session_id": "sess_cnd_001247",
        "assessment_id": "ASS_AI_ENG_20250915_001",
        "websocket_url": "wss://scout-api.studai.one/ws/assessment/sess_cnd_001247",
        "ai_avatar": {
            "avatar_id": "PROF_ALEX_V3",
            "avatar_url": "https://cdn.studai.one/avatars/alex_v3.mp4",
            "voice_model": "neural_voice_v3"
        },
        "first_section": {
            "section_id": "vision_alignment",
            "title": "Vision &amp; Cultural Alignment",
            "estimated_duration": 10,
            "instructions": "We'll start with understanding your career vision..."
        }
    }
    """
```

WebSocket /ws/assessment/{session_id}

```
# Real-time assessment WebSocket connection
@router.websocket("/ws/assessment/{session_id}")
async def assessment_websocket(websocket: WebSocket, session_id: str):
    """
    WebSocket endpoint for real-time assessment interaction

    Message Types:

    1. Question Delivery:
    {
        "type": "question",
        "question_id": "VIS_001",
        "question_text": "Describe your ideal work environment...",
        "question_type": "open_ended",
        "time_limit": 180,
        "multimodal_capture": true
    }

    2. Response Submission:
    {
        "type": "response",
        "question_id": "VIS_001",
        "response_text": "My ideal work environment...",
        "time_taken": 145,
        "metadata": {
            "word_count": 156,
            "typing_pattern": {...},
            "pause_analysis": {...}
        }
    }

    3. Real-time Analysis:
    {
        "type": "analysis_update",
        "question_id": "VIS_001",
        "preliminary_score": 85,
        "insights": {
            "communication_clarity": "excellent",
            "cultural_alignment": "strong_match",
            "confidence_level": "high"
        }
    }

    4. Section Transition:
    {
        "type": "section_complete",
        "completed_section": "vision_alignment",
        "next_section": {
            "section_id": "technical_assessment",
            "title": "Technical Deep Dive",
            "format": "coding_sandbox"
        }
    }
    """
```

4. AI ANALYSIS ENGINE

4.1 Multimodal Analysis

POST /api/v3/assessment/multimodal/analyze

```
# Multimodal response analysis
@router.post("/assessment/multimodal/analyze")
async def analyze_multimodal_response(analysis_request: MultimodalAnalysisRequest):
    """
    Analyze candidate response across multiple modalities

    Request Body:
    {
        "session_id": "sess_cnd_001247",
        "question_id": "AIV_TEC_001",
        "modalities": {
            "text": {
                "response": "In my recent project...",
                "word_count": 245,
                "sentiment": "positive",
                "key_phrases": ["machine learning", "optimization", "user experience"]
            },
            "audio": {
                "audio_url": "https://temp-storage.studai.one/audio/sess_001247_q001.wav'
                "duration": 180,
                "transcript_confidence": 0.95
            },
            "video": {
                "video_url": "https://temp-storage.studai.one/video/sess_001247_q001.mp4'
                "duration": 180,
                "frame_analysis_enabled": true
            }
        }
    }

    Response:
    {
        "analysis_id": "MMA_001247_TEC_001",
        "overall_confidence": 0.92,
        "modality_scores": {
            "verbal_communication": {
                "score": 88,
                "insights": {
                    "clarity": "excellent",
                    "technical_accuracy": "high",
                    "confidence_level": "strong",
                    "speech_pace": "optimal"
                }
            },
            "non_verbal_communication": {
                "score": 85,
                "insights": {
                    "eye_contact": "consistent",
                    "facial_expressions": "engaged",
                    "gesture_usage": "natural",
                    "posture": "professional"
                }
            },
            "content_analysis": {
                "score": 91,
                "insights": {
                    "technical_depth": "advanced",
                    "problem_solving": "systematic",
                    "business_understanding": "strong",
                    "innovation_thinking": "creative"
                }
            }
        },
        "predictive_indicators": {
            "cultural_fit_probability": 0.89,
            "performance_prediction": 0.87,
            "retention_likelihood": 0.84
        }
    }
    """
```

4.2 Behavioral Analysis

POST /api/v3/ai/analyze-behavior

```
# Comprehensive behavioral analysis
@router.post("/ai/analyze-behavior")
async def analyze_candidate_behavior(analysis_request: BehavioralAnalysisRequest):
    """
    Analyze candidate behavior patterns across all assessment interactions

    Request Body:
    {
        "assessment_id": "ASS_AI_ENG_20250915_001",
        "analysis_scope": "comprehensive",
        "include_predictions": true,
        "comparison_cohort": "ai_engineers_senior"
    }

    Response:
    {
        "analysis_id": "BA_001247_COMP",
        "candidate_profile": {
            "communication_style": {
                "score": 88,
                "characteristics": ["clear", "technical", "engaging"],
                "confidence_interval": [85, 91]
            },
            "problem_solving_approach": {
                "score": 91,
                "methodology": "systematic_analytical",
                "creativity_index": 85,
                "persistence_score": 89
            },
            "technical_competency": {
                "score": 90,
                "depth_score": 92,
                "breadth_score": 87,
                "learning_agility": 89
            },
            "cultural_alignment": {
                "score": 89,
                "innovation_mindset": 91,
                "collaboration_preference": 87,
                "growth_orientation": 90
            }
        },
        "predictive_models": {
            "performance_prediction": {
                "score": 87,
                "confidence": 0.89,
                "timeframe": "first_year",
                "factors": ["technical_skills", "cultural_fit", "learning_ability"]
            },
            "retention_prediction": {
                "1_year": 0.92,
                "3_year": 0.85,
                "5_year": 0.78,
                "risk_factors": ["career_growth_pace", "compensation_expectations"]
            },
            "promotion_timeline": {
                "next_level": "18-24 months",
                "confidence": 0.81,
                "development_areas": ["leadership_skills", "cross_functional_experience"]
            }
        }
    }
    """
```

5. REPORTING & ANALYTICS

5.1 Candidate Reports

GET /api/v3/admin/reports/candidate/{candidate_id}

```
# Generate comprehensive candidate report
@router.get("/admin/reports/candidate/{candidate_id}")
async def generate_candidate_report(
    candidate_id: str,
    report_type: str = "comprehensive",
    include_raw_data: bool = False
):
    """
    Generate detailed candidate assessment report

    Response:
    {
        "report_id": "RPT_001247_COMP",
        "generated_at": "2025-09-15T16:30:00Z",
        "candidate": {
            "id": "CND_2025_001247",
            "name": "Arjun Krishnamurthy",
            "position": "Senior AI Engineer",
            "assessment_completed": "2025-09-15T14:32:45Z"
        },
        "executive_summary": {
            "overall_recommendation": "highly_recommended",
            "confidence_score": 0.91,
            "key_strengths": [
                "Exceptional technical depth in AI/ML",
                "Strong problem-solving methodology",
                "Excellent cultural alignment"
            ],
            "development_areas": [
                "Cross-functional leadership experience",
                "Project management frameworks"
            ],
            "summary_text": "Arjun demonstrates exceptional technical competency..."
        },
        "detailed_scoring": {
            "vision_alignment": {
                "score": 92,
                "breakdown": {
                    "cultural_fit": 94,
                    "value_alignment": 90,
                    "communication_clarity": 92
                },
                "insights": ["Strong innovation mindset", "Collaborative approach"]
            },
            "technical_assessment": {
                "score": 90,
                "breakdown": {
                    "coding_proficiency": 92,
                    "system_design": 88,
                    "best_practices": 89,
                    "problem_solving": 91
                }
            }
        },
        "ai_insights": {
            "cultural_fit_analysis": "Excellent alignment with StudAI values...",
            "performance_prediction": "High likelihood of success in senior roles...",
            "career_trajectory": "Technical leadership path within 18-24 months...",
            "retention_forecast": "85% probability of 3+ year retention..."
        },
        "recommendations": {
            "hiring_decision": "strongly_recommend",
            "optimal_role": "Senior AI Engineer - Autonomous Systems",
            "compensation_range": "$140K - $160K",
            "onboarding_focus": [
                "Advanced MLOps practices",
                "Cross-functional collaboration protocols"
            ]
        }
    }
    """
```

5.2 Analytics Dashboard

GET /api/v3/admin/analytics/dashboard

```
# Dashboard analytics data
@router.get("/admin/analytics/dashboard")
async def get_dashboard_analytics(
    date_range: str = "30d",
    include_predictions: bool = True
):
    """
    Retrieve comprehensive dashboard analytics

    Response:
    {
        "period": {
            "start": "2025-08-16T00:00:00Z",
            "end": "2025-09-15T23:59:59Z",
            "range": "30d"
        },
        "overview_metrics": {
            "total_assessments": 347,
            "completed_assessments": 298,
            "completion_rate": 85.9,
            "average_score": 76.8,
            "high_performers": 89,
            "recommended_candidates": 156
        },
        "performance_trends": {
            "daily_completions": [
                {"date": "2025-09-15", "completed": 23, "invited": 28},
                {"date": "2025-09-14", "completed": 19, "invited": 25}
            ]
        },
        "role_analysis": [
            {
                "position": "AI Engineer",
                "total_assessments": 87,
                "average_score": 84.2,
                "completion_rate": 89.7,
                "recommendation_rate": 67.8,
                "top_skills": ["Machine Learning", "Python", "System Design"],
                "trend": "improving"
            }
        ]
    }
    """
```

6. REAL-TIME MONITORING

GET /api/v3/admin/monitoring/live

```
# Real-time assessment monitoring
@router.get("/admin/monitoring/live")
async def get_live_monitoring_data():
    """
    Get real-time assessment monitoring data

    Response:
    {
        "active_sessions": 23,
        "system_status": "healthy",
        "active_assessments": [
            {
                "session_id": "sess_cnd_001250",
                "candidate_name": "Sneha Patel",
                "position": "DevOps Engineer",
                "current_section": "technical_assessment",
                "progress_percentage": 68,
                "time_remaining": 12,
                "performance_indicators": {
                    "engagement_level": "high",
                    "response_quality": "above_average",
                    "technical_accuracy": "excellent"
                }
            }
        ],
        "system_metrics": {
            "api_response_time": "145ms",
            "websocket_connections": 45,
            "database_performance": "optimal",
            "ai_service_latency": "230ms",
            "error_rate": 0.02
        }
    }
    """
```

Background Task Processing

Celery Task Definitions

```
# Background task examples

@celery_app.task
def process_multimodal_analysis(assessment_id: str, modality_data: dict):
    """Process multimodal analysis in background"""
    # Heavy AI processing for voice, video, behavioral analysis
    pass

@celery_app.task
def generate_predictive_models(candidate_id: str):
    """Generate predictive success models"""
    # ML model inference for success prediction
    pass

@celery_app.task
def send_assessment_reminder(candidate_id: str, reminder_type: str):
    """Send automated assessment reminders"""
    # Email/SMS reminder sending
    pass

@celery_app.task
def cleanup_expired_sessions():
    """Clean up expired assessment sessions"""
    # Regular cleanup of old sessions and temporary data
    pass
```

API Rate Limiting & Security

Rate Limiting Configuration

```
# Rate limiting examples
RATE_LIMITS = {
    "admin_endpoints": "1000/hour",
    "candidate_endpoints": "100/hour",
    "ai_analysis": "50/hour",
    "file_upload": "20/hour"
}
```

Security Headers

```
# Security configuration
SECURITY_HEADERS = {
    "X-Content-Type-Options": "nosniff",
    "X-Frame-Options": "DENY",
    "X-XSS-Protection": "1; mode=block",
    "Strict-Transport-Security": "max-age=31536000",
    "Content-Security-Policy": "default-src 'self'"
}
```

Error Handling & Status Codes

Standard HTTP Status Codes

- 200: Success
- 201: Created
- 400: Bad Request
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found
- 422: Validation Error
- 429: Rate Limited
- 500: Internal Server Error

Custom Error Response Format

```
{
    "error": {
        "code": "VALIDATION_ERROR",
        "message": "Invalid request data",
        "details": {
            "field": "email",
            "issue": "Invalid email format"
        },
        "timestamp": "2025-09-15T16:30:00Z",
        "request_id": "req_12345"
    }
}
```

This comprehensive backend API documentation provides the complete technical foundation for implementing the S.C.O.U.T. v3 autonomous hiring intelligence platform with all necessary endpoints, data models, and integration capabilities.