

고객서비스팀 **Agent [JJ-501-104]** 개발 계획서

1. AI Agent 개발 방법 및 절차

1.1 기능 정의

핵심 업무

- 건강보험 가입 및 자격 관련 상담
- 보험료 납부 및 환급 업무
- 건강보험증 발급 및 재발급
- 피부양자 등록 및 변경 업무
- 민원 접수 및 처리
- 건강보험 제도 안내 및 상담
- 다국어 상담 서비스 (영어, 중국어, 일본어)

AI Agent 역할

- 24/7 자동 고객상담 서비스
- 실시간 자격 조회 및 변경 처리
- 보험료 계산 및 납부 안내
- 민원 자동 분류 및 1차 처리
- 다국어 실시간 번역 상담
- 고객 맞춤형 정보 제공

1.2 개발 절차

1. 고객서비스 프로세스 분석 (1일)
2. 다국어 상담 시스템 설계 (2일)
3. 고객 응대 AI 모델 학습 (3일)
4. 옴니채널 서비스 구현 (2일)

2. DeepSeek R1 Fine-tuning 데이터셋

2.1 공개 데이터

- 국민건강보험법 및 관련 법령
- 건강보험 가입 및 보험료 부과 기준
- 피부양자 인정 기준 및 절차
- 건강보험 급여 범위 및 본인부담금
- 고객서비스 표준 매뉴얼
- 건강보험 FAQ 데이터베이스

- 다국어 건강보험 안내 자료
- 민원 처리 표준 절차서

2.2 비공개 데이터

- 제주지사 고객상담 이력 데이터
- 민원 유형별 처리 사례 데이터
- 다국어 상담 대화 로그
- 고객 만족도 조사 결과
- 복잡한 사례의 해결 과정 데이터
- 상담원 교육 자료 및 스크립트
- 지역별 고객 특성 분석 데이터

3. 1주차 개발 일정표

1일차 (월요일) - 고객서비스 프로세스 분석

오전 (09:00-13:00)

고객서비스 체계 분석

- 09:00-09:30: 건강보험 고객서비스 전체 체계 파악
- 09:30-10:30: 상담 유형별 처리 프로세스 매핑
- 10:30-11:00: 휴식
- 11:00-12:00: 민원 분류 체계 및 처리 절차 분석
- 12:00-13:00: 다국어 상담 현황 및 니즈 분석

오후 (14:00-18:00)

AI 자동화 범위 설정

- 14:00-15:00: 자동 처리 가능한 상담 영역 식별
- 15:00-16:00: 인간 상담원 에스컬레이션 기준 정의
- 16:00-16:30: 휴식
- 16:30-17:30: 고객 만족도 향상을 위한 개인화 서비스 설계
- 17:30-18:00: 고객서비스팀 Agent 전체 아키텍처 설계

2일차 (화요일) - 다국어 상담 시스템 설계

오전 (09:00-13:00)

다국어 지원 시스템 설계

- 09:00-10:00: 한국어/영어/중국어/일본어 상담 시스템 설계
- 10:00-11:00: 실시간 번역 및 문화적 맥락 고려 시스템
- 11:00-11:30: 휴식

- 11:30-12:30: 언어별 고객 응대 스타일 차별화 설계
- 12:30-13:00: 제주어 기본 표현 지원 시스템 설계

오후 (14:00-18:00)

고객 상호작용 최적화

- 14:00-15:00: 고객 감정 인식 및 공감 대응 시스템 설계
 - 15:00-16:00: 상담 채널별(전화/채팅/웹/모바일) 최적화 설계
 - 16:00-16:30: 휴식
 - 16:30-17:30: 고객 이력 기반 개인화 서비스 시스템 설계
 - 17:30-18:00: 복잡한 문의 처리를 위한 전문가 연결 시스템 설계
-

3일차 (수요일) - 고객응대 학습 데이터 구축

오전 (09:00-13:00)

상담 시나리오 데이터 생성

- 09:00-10:00: 건강보험 가입/변경 상담 시나리오 데이터 구축
- 10:00-11:00: 보험료 관련 문의 대화 데이터 생성
- 11:00-11:30: 휴식
- 11:30-12:30: 민원 처리 시나리오별 대화 데이터 구축
- 12:30-13:00: 긴급상황 대응 시나리오 데이터 생성

오후 (14:00-18:00)

다국어 및 문화적 맥락 데이터

- 14:00-15:00: 언어별 예의와 존댓말 표현 데이터 구축
 - 15:00-16:00: 문화적 차이를 고려한 설명 방식 데이터
 - 16:00-16:30: 휴식
 - 16:30-17:30: 제주 지역 특성을 반영한 상담 데이터
 - 17:30-18:00: 고객 감정 상태별 대응 방식 데이터 구축
-

4일차 (목요일) - DeepSeek R1 Fine-tuning

오전 (09:00-13:00)

고객서비스 특화 모델 학습

- 09:00-10:00: DeepSeek R1 고객응대 도메인 초기 설정
- 10:00-11:00: 친절하고 정확한 응답 생성 모델 학습
- 11:00-11:30: 휴식
- 11:30-12:30: 다국어 상담 및 번역 품질 향상 학습
- 12:30-13:00: 고객 감정 인식 및 공감 대응 모델 학습

오후 (14:00-18:00)

개인화 서비스 최적화

- 14:00-15:00: 고객 이력 기반 맞춤형 응답 모델 학습
 - 15:00-16:00: 복잡한 문의 처리 및 에스컬레이션 판단 모델
 - 16:00-16:30: 휴식
 - 16:30-17:30: 상담 품질 및 고객 만족도 예측 모델 학습
 - 17:30-18:00: 모델 성능 검증 및 최적화
-

5일차 (금요일) - Agent 구현 및 서비스 통합

오전 (09:00-13:00)

고객서비스팀 Agent 핵심 기능 구현

- 09:00-10:00: 실시간 고객상담 자동응답 모듈 구현
- 10:00-11:00: 다국어 번역 및 문화적 적응 모듈 구현
- 11:00-11:30: 휴식
- 11:30-12:30: 민원 자동 분류 및 처리 모듈 구현
- 12:30-13:00: 고객 개인화 서비스 모듈 구현

오후 (14:00-18:00)

옴니채널 서비스 통합

- 14:00-15:00: 상위 Agent [JJ-501]와의 연동 구현
 - 15:00-16:00: 다양한 채널(웹/모바일/전화/채팅) 통합 테스트
 - 16:00-16:30: 휴식
 - 16:30-17:30: 고객 만족도 실시간 모니터링 시스템 구현
 - 17:30-18:00: 전체 시스템 성능 최적화 및 문서화
-

주요 기능 모듈

1. 고객상담 자동응답 모듈

```
class CustomerServiceBot:
```

```
    def process_customer_inquiry(self, inquiry):
        # 고객 문의 자동 분류 및 응답
        inquiry_type = self.classify_inquiry_type(inquiry)
        response = self.generate_contextual_response(inquiry, inquiry_type)
        satisfaction_prediction = self.predict_customer_satisfaction(response)

        if satisfaction_prediction < self.escalation_threshold:
            return self.escalate_to_human_agent(inquiry)
```

```
return response
```

```
def handle_complex_case(self, complex_inquiry):  
    # 복잡한 문의 처리 및 전문가 연결  
    complexity_score = self.assess_inquiry_complexity(complex_inquiry)  
    if complexity_score > self.complexity_threshold:  
        return self.route_to_specialist(complex_inquiry)  
    return self.process_with_enhanced_context(complex_inquiry)
```

2. 다국어 상담 모듈

```
class MultilingualSupport:  
    def provide_multilingual_service(self, inquiry, source_language):  
        # 다국어 상담 서비스 제공  
        detected_language = self.detect_language(inquiry)  
        if detected_language != 'korean':  
            translated_inquiry = self.translate_to_korean(inquiry, detected_language)  
            korean_response = self.generate_response(translated_inquiry)  
            final_response = self.translate_to_target_language(korean_response,  
detected_language)  
            return self.adapt_cultural_context(final_response, detected_language)  
        return self.generate_response(inquiry)  
  
    def handle_jeju_dialect(self, jeju_inquiry):  
        # 제주어 기본 표현 처리  
        standard_korean = self.convert_jeju_to_standard(jeju_inquiry)  
        return self.generate_response(standard_korean)
```

3. 개인화 서비스 모듈

```
class PersonalizedService:  
    def provide_personalized_service(self, customer_id, inquiry):  
        # 고객 맞춤형 서비스 제공  
        customer_profile = self.get_customer_profile(customer_id)  
        service_history = self.get_service_history(customer_id)  
  
        personalized_response = self.generate_personalized_response(  
            inquiry, customer_profile, service_history  
        )  
  
        # 고객 관심사 기반 추가 정보 제공  
        relevant_info = self.suggest_relevant_services(customer_profile)  
  
        return {  
            'response': personalized_response,  
            'additional_services': relevant_info,  
            'satisfaction_score': self.predict_satisfaction(personalized_response)
```

```
}
```

```
def track_customer_journey(self, customer_id):  
    # 고객 여정 추적 및 최적화  
    journey_data = self.analyze_customer_touchpoints(customer_id)  
    improvement_suggestions = self.identify_journey_improvements(journey_data)  
    return improvement_suggestions
```

4. 민원 관리 모듈

```
class ComplaintManagement:  
    def process_complaint(self, complaint_data):  
        # 민원 자동 분류 및 처리  
        complaint_category = self.classify_complaint(complaint_data)  
        urgency_level = self.assess_urgency(complaint_data)  
  
        if urgency_level == 'high':  
            return self.initiate_immediate_response(complaint_data)  
  
        resolution_plan = self.generate_resolution_plan(complaint_category)  
        return {  
            'category': complaint_category,  
            'resolution_plan': resolution_plan,  
            'estimated_resolution_time': self.estimate_resolution_time(complaint_category)  
        }  
  
    def monitor_complaint_resolution(self, complaint_id):  
        # 민원 처리 과정 모니터링  
        progress = self.track_resolution_progress(complaint_id)  
        customer_updates = self.generate_progress_updates(progress)  
        return customer_updates
```

5. 품질 관리 모듈

```
class ServiceQualityManager:  
    def monitor_service_quality(self, interaction_data):  
        # 서비스 품질 실시간 모니터링  
        quality_metrics = self.calculate_quality_metrics(interaction_data)  
        areas_for_improvement = self.identify_improvement_areas(quality_metrics)  
  
        if quality_metrics['satisfaction_score'] < self.quality_threshold:  
            return self.trigger_quality_improvement_process(interaction_data)  
  
        return quality_metrics  
  
    def generate_service_insights(self, period_data):  
        # 서비스 인사이트 및 개선 제안 생성
```

```
trends = self.analyze_service_trends(period_data)
recommendations = self.generate_improvement_recommendations(trends)
return {
    'service_trends': trends,
    'improvement_recommendations': recommendations,
    'performance_indicators': self.calculate_kpis(period_data)
}
```

고객서비스 품질 보장

1. 서비스 품질 지표

- 응답 정확도: **96%** 이상
- 고객 만족도: **4.6/5.0** 이상
- 첫 응답 해결률: **85%** 이상
- 평균 응답 시간: **10초** 이내
- 다국어 번역 품질: **94%** 이상

2. 고객 보호 장치

- 프라이버시 보호: 개인정보 암호화 및 최소 수집 원칙
- 공정한 서비스: 차별 없는 동등한 서비스 제공
- 투명성: AI 응답 근거 및 한계 명시
- 에스컬레이션: 복잡한 사안의 신속한 전문가 연결

지속적 개선 체계

1. 피드백 수집 및 분석

- 실시간 만족도 조사: 상담 완료 후 즉시 평가
- 고객 의견 분석: 텍스트 마이닝을 통한 개선점 도출
- 서비스 **Gap** 분석: 기대 대비 실제 서비스 수준 측정

2. 모델 업데이트 및 개선

- 주간 성능 리뷰: 주요 지표 점검 및 조정
- 월간 모델 업데이트: 새로운 사례 학습 및 성능 향상
- 분기별 대규모 개선: 아키텍처 개선 및 신기능 추가

보안 및 규정 준수

1. 개인정보 보호

- 데이터 암호화: 고객 정보 **AES-256** 암호화

- 접근 제어: 역할 기반 접근 권한 관리
- 로그 관리: 모든 상담 내역 안전한 보관 및 추적

2. 서비스 연속성

- **24/7** 무중단 서비스:고가용성 시스템 구축
- 재해 복구: 백업 시스템 및 복구 절차 완비
- 부하 분산: 트래픽 급증 시 자동 확장

성과 예측 및 기대효과

1. 정량적 효과

- 상담 처리량: **300%** 증가
- 대기시간: **90%** 감소
- 운영비용: **40%** 절감
- 고객 만족도: **15%** 향상

2. 정성적 효과

- 서비스 접근성: **24시간** 언제든지 상담 가능
- 언어 장벽 해소: 다국어 실시간 지원
- 개인화 서비스: 고객별 맞춤 정보 제공
- 일관된 서비스: 상담원별 편차 최소화

다음 단계 계획

- 전체 통합 테스트: 4개 팀 **Agent** 연동 검증
- 실제 환경 파일럿: 제주지사 일부 업무 시범 적용
- 성능 최적화: 실사용 데이터 기반 지속적 개선
- 전국 확산 준비: 다른 지역 적용을 위한 표준화