

creative
FDS

By 주영성, 고동연, 강슬기, 김용혁

목차

01 기획의도 및 배경

02 목표

03 기대효과

04 제안의 특장점

05 주요기능

06 개발방법론

07 마인드 맵

08 FDS 요인 분석

- 계좌 이체
- 해외 카드 결제
- 사망 보험
- 자동차 보험

09 논리설계

- ERD

10 물리설계

- 테이블 정의서

11 시스템 아키텍쳐

12 기능명세서(FBS)

13 스토리보드

14 소스코드

15 일정(WBS)

16 조직도(WBS)

17 사용된 프로그램

1. 기획의도 및 배경

계좌 이체

최근 들어 수많은 방식의 보이스 피싱이 발생하고 있습니다. 예를 들어, 검사를 사칭하여 돈을 인출하게 만들어 현금을 주고 받는 방법, 가족이나 지인을 사칭하여 계좌 송금을 유도하는 방식 등 여러 방식의 보이스 피싱이 성행하고 있습니다. FDS를 도입하여 보이스 피싱을 탐지하고 예방하여 고객들의 자산을 보호하고 금융기관의 신뢰도를 높일 수 있습니다.

해외 카드 결제

현대 사회에서 해외여행, 글로벌 전자상거래, 해외유학 등의 증가로 인해 해외 카드 결제는 지속적으로 증가하고 있습니다. 이에 따라 해외에서의 카드 결제는 카드사와 고객 모두에게 중요한 서비스가 되었습니다. 그러나, 이러한 결제의 증가와 함께 카드 사기 위험도 높아지고 있습니다. 해외 카드 결제에서 발생하는 사기 행위는 고객과 카드사에 큰 재정적 피해를 입힙니다. 특히, 비정상적이거나 의심스러운 거래를 사전에 탐지하지 못하면 고객 신뢰도 저하, 브랜드 이미지 손상, 직접적인 재정 손실 등의 부정적 결과를 초래할 수 있습니다.

1. 기획의도 및 배경

사망 보험

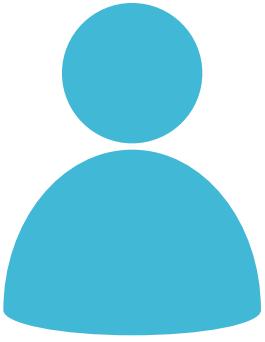
사망보험은 보험금을 지급하는 과정에서 부정행위의 위험이 있습니다. 예를 들어, 보험 사기, 적발되지 않은 건강 상태의 은폐 등이 해당됩니다. FDS를 도입하여 이러한 부정행위를 탐지하고 예방함으로써 보험사의 손실을 최소화할 수 있습니다. 또한 보험 사기는 금융 기관의 신뢰도를 훼손 시킬 수 있습니다. FDS를 도입하여 부정행위를 탐지하고 처리함으로써 금융 기관의 신뢰도를 높일 수 있습니다.

자동차 보험

자동차보험 FDS는 보험 사기를 효과적으로 탐지하고 예방하여 보험사와 고객 모두의 재정적 손실을 줄이는 것을 목표로 두고 있습니다. 이를 통해 보험 가입자들이 공정하고 투명한 환경에서 혜택을 받을 수 있도록 하며, 보험사의 신뢰성을 높여 전체 보험 산업의 건전성을 강화하고자 합니다

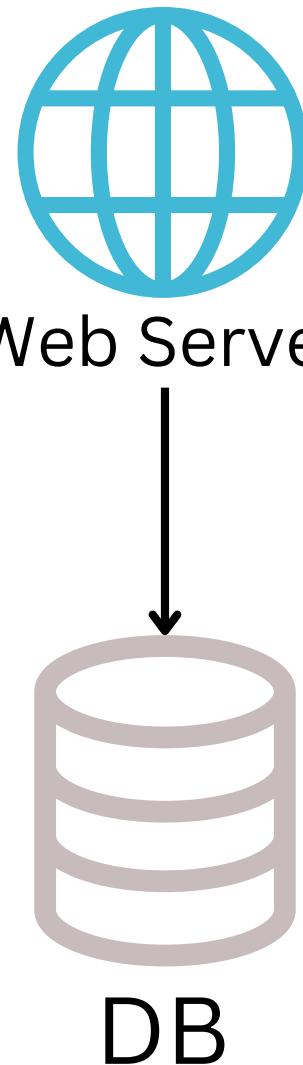
2. 목표

정보 수집



- 사용자 정보
- 카드 정보
- IP 정보

서버로 정보 수신 및 저장



통계 및 분석



- 거래 정보 매핑
- 연간 로그 분석
- 패턴 분석
- 통계 추출

대응

(이상거래에 대한 대응)



- 이상 거래 탐지
- 룰 패턴 검사
- 리포트 시스템

3. 기대효과



부정활동을 식별하여
회피하고 예방



비즈니스의 안전성과
신뢰성을 향상시키고
운영 효율성 증가



재정적 손실을 줄이고
고객 신뢰를 유지
규정 준수에 대한 위험도 감소

4. 제안의 특장점

FDS



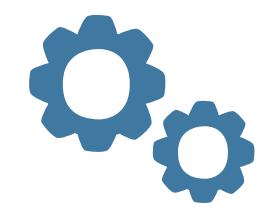
데이터 수집,
가공, 분석
종합 패키지
시스템 구축

통계 기반의
데이터 제공

약 30여개의
다양한
시나리오(Rule)
보유

FDS 구축 표준
가이드 준수

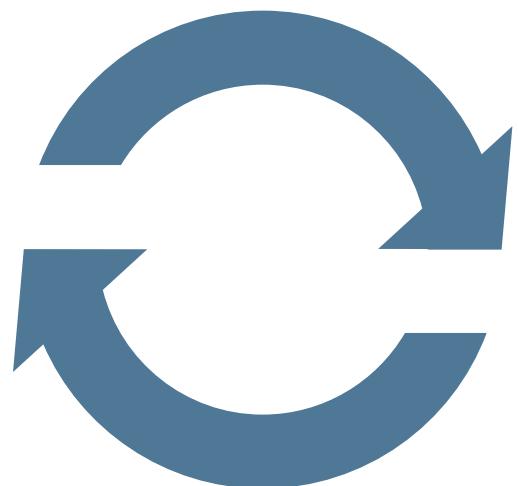
5. 주요기능



Rule / 통계 기반의 이상거래탐지 시스템



chart.js를 이용한 데이터 시각화



ajax를 이용한 비동기적 페이지 전환

검색, 페이지 등 사용자 친화적 UX/UI

6. 개발방법론

PHASE 1
구축 시스템 청사
진 제시

PHASE 2
목표 업무 분석

PHASE 3
목표 업무 설계

PHASE 4
목표 업무 개발

PHASE 5
목표 업무 이행

요구사항 분석 및
현행 시스템 기능 분석

개발 환경구성

목표 기능 모델링

A/P 코딩

시범 운영

시스템 청사진 제시

개발 목표 업무 기능
상세요구 사항 분석

목표 기능 기능설계

데이터 전환

개발 완료된 시스템

개발 범위 확정

개발 범위의
상세합의 도출

상세설계 및
SPEC 작성

매뉴얼 작성

실행에 따른
DRP 검토

개발 계획의 상세화

목표 기능 구성도

ERD, DB SCHEMA

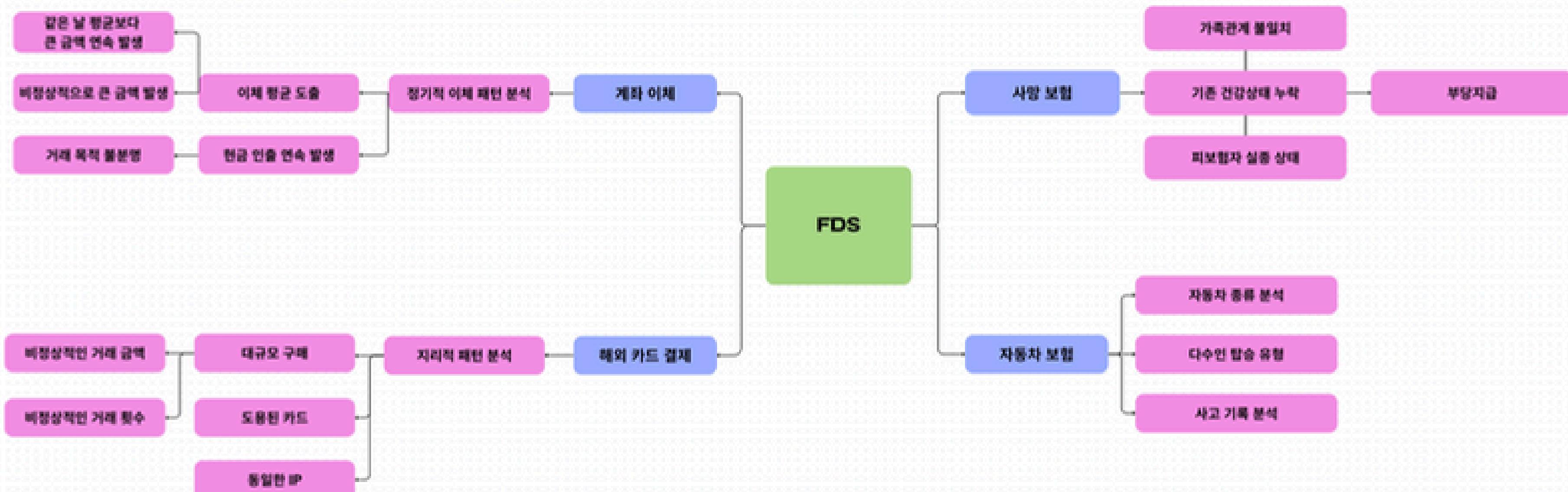
단위 테스트 병행

완료 보고서

품질 관리 계획 수립

품질 보증 활동 → 주기적인 품질 **REVIEW MEETING**
변경 관리, 표준 관리 등

7. 마인드맵



8. FDS 요인 분석

• 해외 카드 결제

번호	요인	해결 방안
1	지리적 패턴 분석	<ul style="list-style-type: none">고객의 결제 지역과 해당 카드의 발급 국가를 비교하여 불일치가 발생할 경우
2	대규모 구매	<ul style="list-style-type: none">일반적으로 작은 금액의 결제를 자주 하는데, 갑자기 큰 금액의 결제가 빈번하게 발생하는 경우
		<ul style="list-style-type: none">하루에 이루어질 수 있는 최대 거래 횟수를 설정하고 이를 초과할 경우
3	도용된 카드	<ul style="list-style-type: none">도용된 카드 정보를 사용하여 해외에서 결제를 시도하는 경우
4	동일한 IP 분석	<ul style="list-style-type: none">여러 사람이 동일한 IP를 차단한 경우 모든 사람에게 해당 IP차단

• 계좌 이체

번호	요인	해결 방안
1	정기적인 이체 패턴 분석	<ul style="list-style-type: none">고객의 계좌에서 이루어지는 이체의 빈도 및 패턴을 분석하여 이상한 활동이 감지될 경우 경고를 발생시킴.특정 기간 동안 매우 빈번하게 거래가 이루어지다가 갑자기 거래가 중단되는 패턴

8.FDS 요인 분석

1	정기적인 이체 패턴 분석	<ul style="list-style-type: none">거래 금액이 통계적으로 비정상적으로 높은 경우를 탐지동일 금액이 반복적으로 입금되고 출금되는 패턴
2	이체 금액 비교	<ul style="list-style-type: none">평소보다 큰 금액이 발생하는 경우이전 이체 이력과 비교하여 급격한 금액 변동이 감지평균적으로 적은 금액이 발생하는 장소에서 큰 금액이 발생한 경우
3	거래 목적 불분명	<ul style="list-style-type: none">입출금 내역이 빈번하지만, 거래 목적이 명확하지 않은 경우
4	사기 거래 여부	<ul style="list-style-type: none">상대방의 계좌가 사기 의심 계좌인지, 사기 계좌로 등록되어있는지 확인

• 사망 보험

번호	요인	해결 방안
1	연령 불일치	<ul style="list-style-type: none">사망 보험 청구자의 연령이 사망 보험 계약 시 기재된 연령과 크게 다른 경우.
2	빠른 보험 청구	<ul style="list-style-type: none">피보험자의 사망 시기가 보험 가입을 한지 얼마 안된 기간에 일어난 경우피보험자의 사망 시기가 보험 가입을 한지 얼마 안된 기간에 일어난 경우

8.FDS 요인 분석

2	빠른 보험 청구	<ul style="list-style-type: none">사망통지가 나오기 전 청구서를 제출한 경우
3	기존 건강 상태 누락	<ul style="list-style-type: none">보험 가입자가 사망 전에 장기적인 건강 문제를 갖고 있었지만 그 내용이 청구서에 없는 경우
4	가족관계 변조	<ul style="list-style-type: none">보험 청구자가 피보험자와 실제로는 가족 관계가 없는 사람임에도 불구하고, 그들을 가족으로 속이고 사망 보험을 청구하는 경우
5	감염병으로 인한 사망	<ul style="list-style-type: none">코로나19와 같은 감염병으로 인한 사망으로 보험을 청구하는 경우
6	해외여행도중 실종신고후 사망 보고	<ul style="list-style-type: none">부부 또는 치매 병력이 있는 조부모와 함께 해외여행도중 현지에서 실종신고 접수후 귀국해 사망 신고를 한 뒤 보험을 청구하는 경우
7	자살로 인한 사망	<ul style="list-style-type: none">정신질환과 같은 질병 기록이 있는 경우

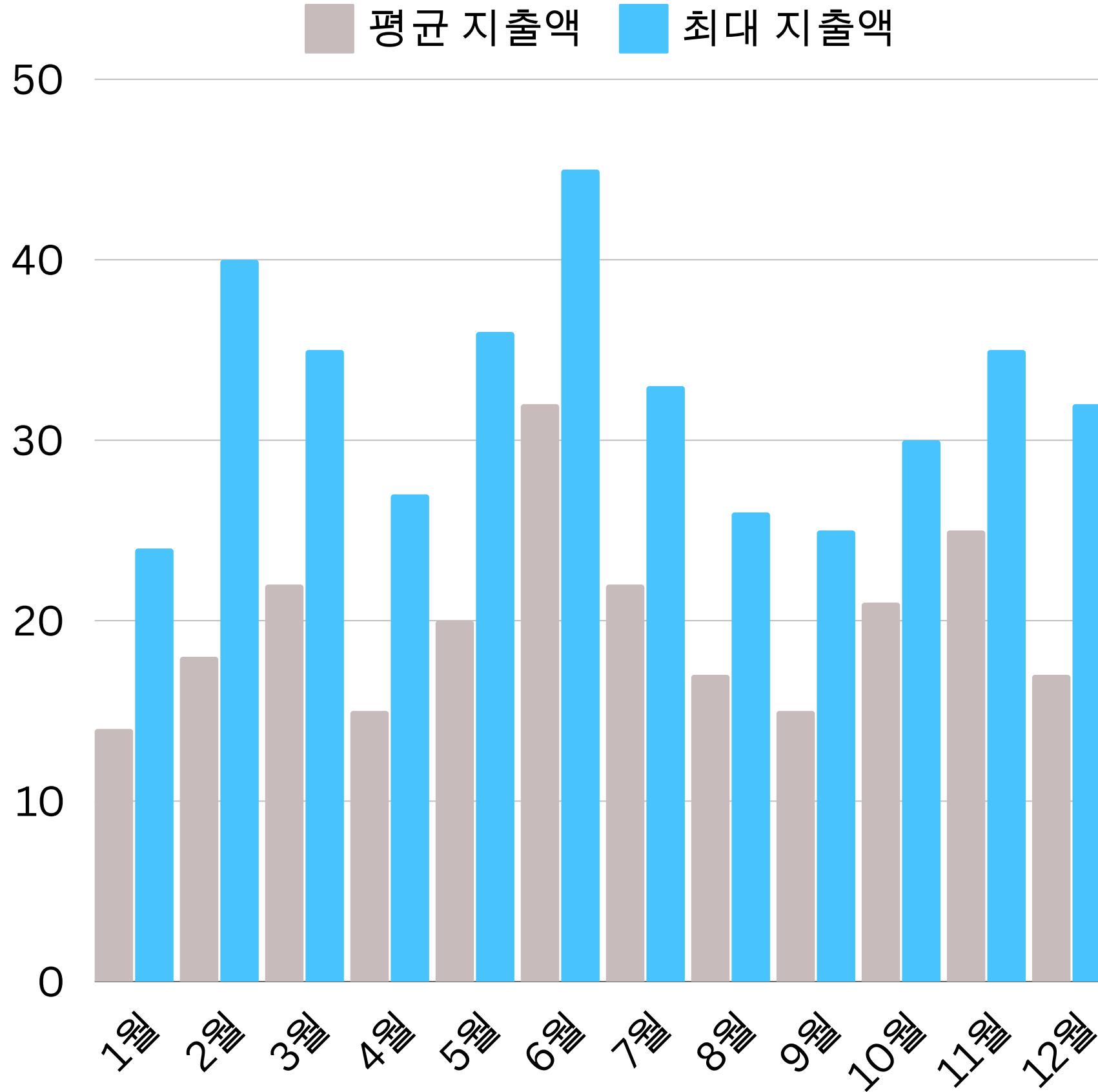
• 자동차 보험

번호	요인	해결 방안
1	다수인 탑승 유형	<ul style="list-style-type: none">차량에 일정 기준 이상의 인원이 탑승해 있을 경우 지급 보류
		<ul style="list-style-type: none">차량의 정원보다 많은 인원이 탑승해 있을 경우 지급 보류
2	자동차종류 분석	<ul style="list-style-type: none">7인승 이상의 차량의 차량 사고 시 지급 보류

8.FDS 요인 분석

3	사고기록분석	<ul style="list-style-type: none">• 비슷한 사고 경위일 경우 지급 보류• 자동차 사고가 짧은 기간안에 재발할 경우 지급 보류
---	--------	--

▣ 해외 카드 결제



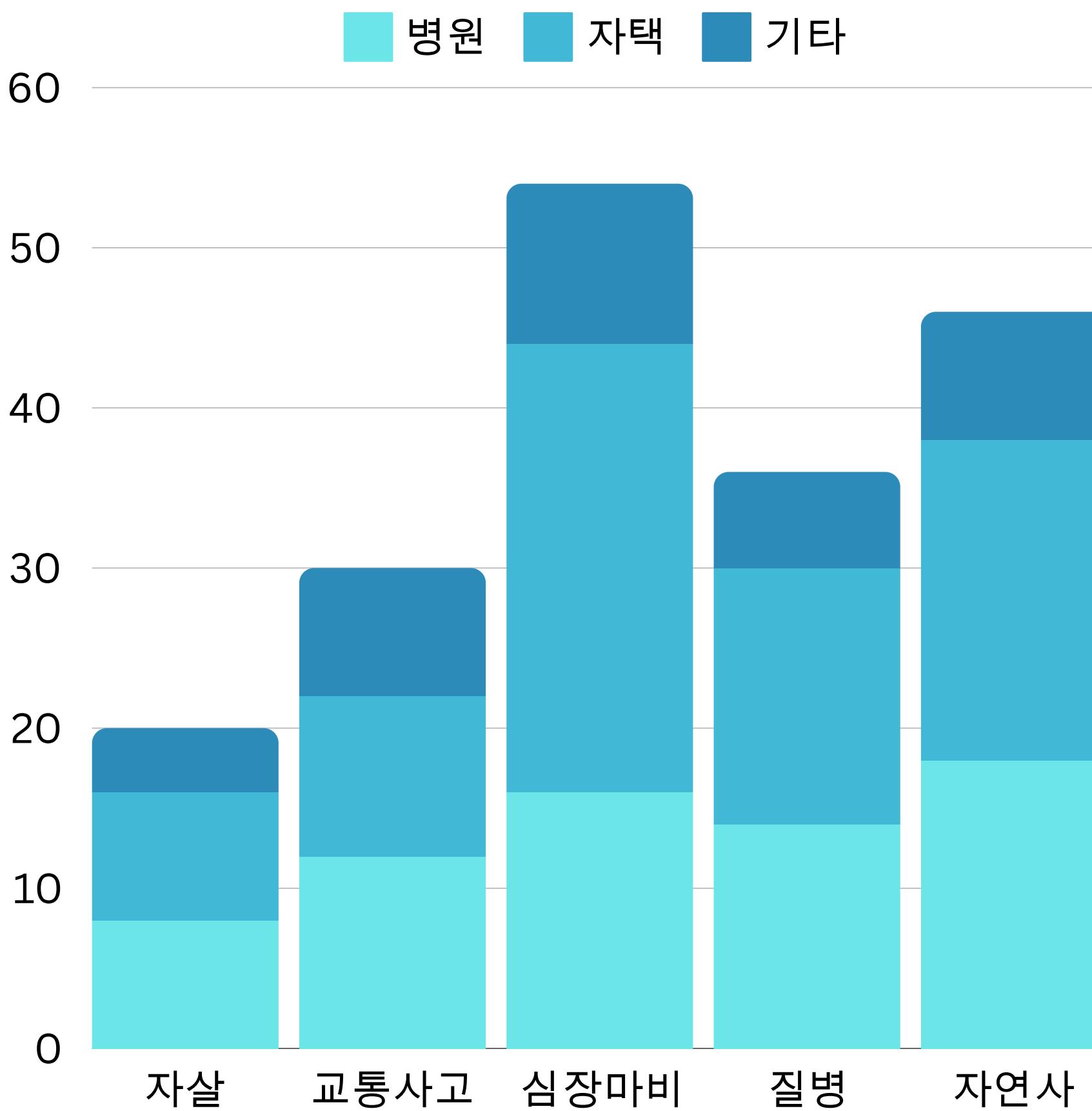
비정상적인 거래 패턴 분석

고객의 소비 패턴 분석

통계 기반의 데이터 제공

사망보험

청구에 대한 부당지급 확인
사망원인 추이 분석
통계 기반의 데이터 제공



9. 논리설계

• ERD



10. 물리설계

• 테이블 정의서

엔티티타입명	고객						
테이블명	CUSTOMERS						
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY
1	customer_id	고객고유번호	일련번호	VARCHAR	40	NOT NULL	PK
2	customer_name	고객이름	이름	VARCHAR	50	NULL	
3	customer_email	이메일	이메일	VARCHAR	100	NULL	
4	customer_phone	연락처	연락처	VARCHAR	13	NULL	
5	customer_addr	주소	주소	VARCHAR	255	NULL	
6	customer_addr_detail	상세주소	주소	VARCHAR	255	NULL	
7	customer_post	우편번호	숫자	NUMBER		NULL	
8	customer_country	거주국가	국가	VARCHAR	2	NULL	

10. 물리설계

• 테이블 정의서

IP등록							
엔티티타입명							
테이블명	IP_AGREE						
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY
1	customer_id	고객고유번호	일련번호	VARCHAR	40	NOT NULL	PK,FK
2	ip_address	IP주소	숫자	NUMBER		NULL	
3	agree_date	IP등록날짜	날짜	DATE		NULL	
IP차단							
엔티티타입명							
테이블명	IP_BLOCK						
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY
1	customer_id	고객고유번호	일련번호	VARCHAR	40	NOT NULL	PK,FK
2	ip_address	IP주소	숫자	NUMBER		NULL	
3	block_date	IP차단날짜	날짜	DATE		NULL	

10. 물리설계

• 테이블 정의서

카드							
CARD							
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY
1	card_id	카드고유번호	일련번호	VARCHAR	40	NOT NULL	PK
2	customer_id	고객고유번호	일련번호	VARCHAR	40	NOT NULL	PK,FK
3	card_num	카드번호	카드번호	VARCHAR	20	NULL	
4	expiry_date	카드등록날짜	날짜	DATE		NULL	
5	issue_country	발급국가	국가	VARCHAR	20	NULL	
6	ERROR_COUNT	거래거절횟수	숫자	NUMBER		NULL	
거래 정지 카드							
TRADING_HALT							
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY

10. 물리설계

• 테이블 정의서

1	card_id	카드고유번호	일련번호	VARCHAR	40	NOT NULL	PK,FK
2	customer_id	고객고유번호	일련번호	VARCHAR	40	NOT NULL	PK,FK
3	halt_date	거래정지날짜	날짜	DATE		NULL	
엔티티타입명	결제						
테이블명	PURCHASE						
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY
1	purchase_id	결제고유번호	일련번호	VARCHAR	40	NOT NULL	PK
2	card_id	카드고유번호	일련번호	VARCHAR	40	NOT NULL	PK,FK
3	customer_id	고객고유번호	일련번호	VARCHAR	40	NOT NULL	PK,FK
4	country_id	국가고유번호	일련번호	VARCHAR	40	NOT NULL	PK,FK
5	purchase_price	결제금액	숫자	NUMBER		NULL	
6	purchase_date	결제날짜	날짜	DATE		NULL	

10. 물리설계

• 테이블 정의서

6	purchase_status	결제상태	상태	VARCHAR	20	NULL	
엔티티타입명	국가						
테이블명	COUNTRY						
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY
1	country_id	국가고유번호	일련번호	VARCHAR	40	NOT NULL	PK
2	country_code	국가코드	국가	VARCHAR	2	NULL	
3	country_name	국가이름	이름	VARCHAR	50	NULL	
4	ip_start_num	IP시작번호	숫자	NUMBER		NULL	
5	ip_end_num	IP끝번호	숫자	NUMBER		NULL	
엔티티타입명	고객						
테이블명	ACCMEMBERS						
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY

10. 물리설계

• 테이블 정의서

1	accmem_num	고객고유번호	일련번호	VARCHAR	40	NOT NULL	PK
2	accmem_name	고객이름	이름	VARCHAR	50	NULL	
3	accmem_addr	주소	주소	VARCHAR	255	NULL	
4	accmem_detail	상세주소	주소	VARCHAR	255	NULL	
5	accmem_post	우편번호	번호	NUMBER		NULL	
6	accmem_tel	연락처	연락처	VARCHAR	13	NULL	
7	accmem_birth	생년월일	날짜	DATE		NULL	
엔티티타입명	고객계좌						
테이블명	ACCOUNT						
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY
1	account_num	계좌고유번호	일련번호	VARCHAR	40	NOT NULL	PK
2	accmem_num	고객고유번호	일련번호	VARCHAR	40	NOT NULL	PK,FK

10. 물리설계

• 테이블 정의서

3	account_date	계좌개설일	날짜	DATE		NULL	
엔티티타입명	입출금						
테이블명	TRANSFERINFO						
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY
1	account_num	계좌고유번호	일련번호	VARCHAR	40	NOT NULL	PK,FK
2	accmem_num	고객고유번호	일련번호	VARCHAR	40	NOT NULL	PK,FK
3	transfer_num	입출금고유번호	일련번호	VARCHAR	40	NOT NULL	PK
4	income_price	입금금액	숫자	NUMBER		NULL	
5	income_date	입금날짜	날짜	DATE		NULL	
6	outcome_price	출금금액	숫자	NUMBER		NULL	
7	outcome_date	출금날짜	날짜	DATE		NULL	
8	transfer_content	입출금내용	내용	VARCHAR	2000	NULL	

10. 물리설계

• 테이블 정의서

피보험자							
엔티티타입명							
테이블명	INSURED						
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY
1	ins_id	피보험자고유번호	일련번호	VARCHAR	40	NOT NULL	PK
2	ins_name	피보험자 이름	이름	VARCHAR	50	NULL	
3	ins_birth	생년월일	날짜	DATE		NULL	
4	ins_addr	주소	주소	VARCHAR	255	NULL	
5	ins_addr_detail	상세주소	주소	VARCHAR	255	NULL	
6	ins_post	우편번호	숫자	NUMBER		NULL	
7	ins_phone	연락처	연락처	VARCHAR	13	NULL	
8	ins_email	이메일	이메일	VARCHAR	100	NULL	
9	ins_case_history	병력	병력	VARCHAR	100	NULL	

10. 물리설계

• 테이블 정의서

10	ins_beneficiary	수령인	이름	VARCHAR	50	NULL	
11	ins_privity	수령인 관계	상태	VARCHAR	20	NULL	
12	ins_ben_phone	수령인 연락처	연락처	VARCHAR	13	NULL	
13	ins_contract_date	보험 계약일	날짜	DATE		NULL	
엔티티타입명	사망보험청구서						
테이블명	DEATH_CLAIM						
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY
1	claim_num	청구고유번호	일련번호	VARCHAR	40	NOT NULL	PK
2	ins_id	피보험자고유번호	일련번호	VARCHAR	40	NOT NULL	PK,FK
3	claim_content	청구내용	내용	VARCHAR	2000	NULL	
4	claim_name	청구인 성명	이름	VARCHAR	50	NULL	
5	claim_birth	청구인 생년월일	날짜	DATE		NULL	

10. 물리설계

• 테이블 정의서

10. 물리설계

• 테이블 정의서

번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY
11	mem_id	회원고유번호	일련번호	VARCHAR	40	NOT NULL	PK
12	mem_name	회원이름	이름	VARCHAR	50	NULL	
13	mem_birth	회원생년월일	날짜	DATE		NULL	
4	mem_car	회원차종	이름	NUMBER		NULL	
5	mem_carnumber	차량번호	일련번호	□VARCHAR	40	NULL	
6	mem_addr	주소	주소	VARCHAR	255	NULL	
4	mem_addrdetail	상세주소	주소	VARCHAR	255	NULL	
5	mem_postcode	우편번호	숫자	NUMBER		NULL	
6	mem_start_date	보험 시작일	날짜	DATE		NULL	
5	mem_end_date	보험 종료일	날짜	DATE		NULL	
6	mem_phone	연락처	연락처	VARCHAR	13	NULL	

10. 물리설계

• 테이블 정의서

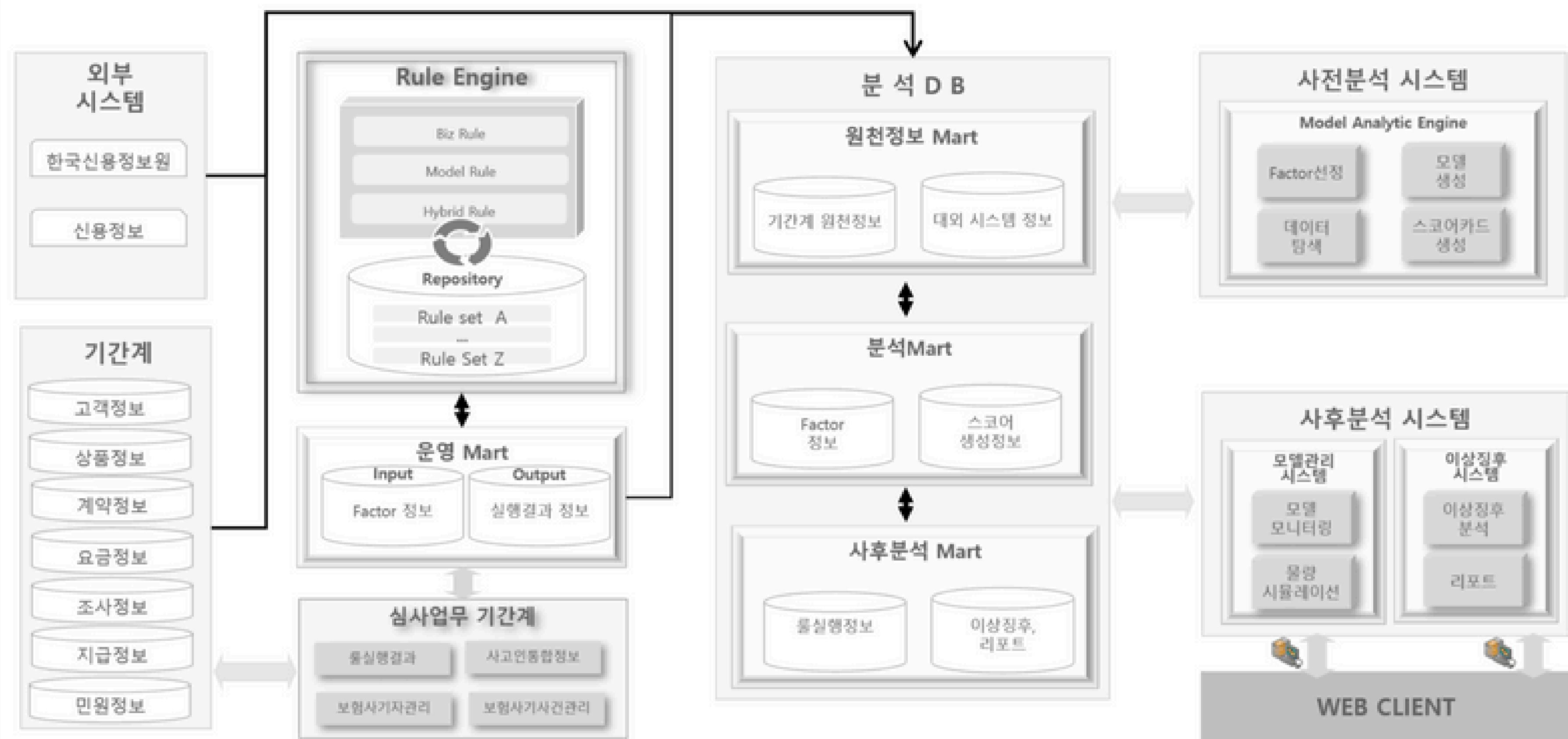
10	mem_email	이메일	이메일	VARCHAR	100	NULL	
11	mem_licensenumber	면허번호	일련번호	VARCHAR	40	NULL	
12	mem_licensetype	면허종류	상태	VARCHAR	20	NULL	
엔티티타입명	자동차보험청구서						
테이블명	CARCLAIM						
번호	컬럼명	속성명	도메인	데이터타입	길이	NULL여부	KEY
1	claim_number	청구서고유번호	일련번호	VARCHAR	40	NOT NULL	PK
2	mem_carnumber	차량번호	일련번호	VARCHAR	40	NOT NULL	PK,FK
3	claim_name	피보험자이름	이름	VARCHAR	50	NULL	
4	claim_car	피보험자차종	이름	VARCHAR	50	NULL	
5	claim_tel	피보험자연락처	연락처	VARCHAR	13	NULL	
6	claim_licensenumber	면허번호	숫자	NUMBER		NULL	

10. 물리설계

• 테이블 정의서

7	claim_licensetype	면허종류	상태	VARCHAR	20	NULL	
8	claim_accidentdate	사고일시	날짜	DATE		NULL	
9	claim_location	사고장소	주소	VARCHAR	255	NULL	
10	claim_situation	사고경위	내용	VARCHAR	2000	NULL	
11	claim_count	탑승자인원수	숫자	NUMBER		NULL	
12	claim_hospital	치료병원	이름	VARCHAR	50	NULL	
13	claim_signname	청구인성명	이름	VARCHAR	50	NULL	
14	claim_addr	청구인주소	주소	VARCHAR	255	NULL	

11. 시스템 아키텍쳐



12. 기능명세서(FBS)

어플리케이션	페이지명	파일명	요소	상세내용	개발	테스트
해외카드결제	카드결제페이지	creditCardForm.html	a	관리자페이지로 이동		
			form	결제정보 입력 시 FDS시나리오를 거친 후, DB에 INSERT함		
			input-text	결제정보에 필요한 결제ID, 고객ID, 카드번호, 유효기간, IP주소를 입력		
			input-number	고객이 결제할 결제금액을 입력		
			input-submit	'제출'버튼 클릭 시 form태그 실행		
	관리자페이지	pgAdmin.html	table	DB에서 가져온 이상거래 리스트를 반복해서 출력		
고객상세페이지	고객상세페이지	customerPage.html	a	결제, 카드, 고객의 고유번호를 적용하여 클릭 시 고객의 상세정보 페이지로 이동		
			a	index페이지로 이동		
			table	최근 1년 매달간의 고객의 총 소비금액, 평균 지출액, 거래횟수를 그래프로 표시. 최근 1달 매일간의 고객의 거래 빈도수를 그래프로 표시.		
			table	해당 고객의 이상거래패턴을 리스트를 반복하여 출력		
			button	비정상적인 금액에 대한 고객과의 연락을 취한 후, 결제 승인 혹은 취소를 결정		

12. 기능명세서(FBS)

애플리케이션	페이지명	파일명	요소	상세내용	개발	테스트
계좌조회	계좌조회 페이지	accountForm.html	table	고객 정보를 입력하는 입력창들을 모아둔 틀		
			table	입력 버튼을 모아둔 틀		
			form	고객 정보를 검색		
			button	뒤로가기 버튼		
	조회결과 페이지	accountResult.html	h3	결과 페이지 제목		
			table	고객의 상세정보를 출력		
			span	고객의 지급 사유에 맞게 출력		
			button	홈으로 링크를 걸어 이동		
			button	통계확인 페이지로 이동		

12. 기능명세서(FBS)

애플리케이션	페이지명	파일명	요소	상세내용	개발	테스트
사망보험	사망보험 관리자 페이지	deathClaimChart.html	a	네비게이션 페이지 이동		
			table	통계 및 청구 추이 분석 내용 출력		
	청구서 조회 페이지	deathClaimForm.html	a	네비게이션 페이지 이동		
			button	청구서 가져오기 버튼		
			form	Rule 기반 시나리오, DB 조회		
			input-text	이름, 내용, 주소, 수령 계좌, 이메일 입력		
			select	사망 원인 및 장소 선택		
			input-submit	확인 버튼 클릭시 form 태그 실행		
	청구서 리스트 페이지	deathClaimList.html	a	네비게이션 페이지 이동		
			form	청구서 검색		
	지급 조회 결과 페이지	deathClaimResult.html	table	시나리오에 의해 지급에 대한 결과 내용을 출력		

12.기능명세서(FBS)

어플리케이션	페이지명	파일명	요소	상세내용	개발	테스트
자동차보험	청구서 입력 페이지	carClaimForm.html	a	자동차보험 연간 청구수 페이지로 이동		
			button	'홈으로' 클릭시 index페이지로 이동		
			form	피보험자가 작성한 정보를 fds 시나리오를 걸친 후 db 입력 or 보류		
			input-text	차량번호, 이름, 차종, 연락처, 면허번호, 면허종류, 사고일시 등 청구인의 정보 입력		
			input-submit	확인 버튼 클릭시 form 태그 실행		
	지급 여부 결과 페이지	carClaimResult.html	table	시나리오에 의한 지급 여부 결정 여부 및 내용을 출력		
자동차보험 연간 청구수 페이지	carClaimChart.html		button	'뒤로가기' 클릭시 청구서 입력 페이지로 이동		
			button	'홈으로' 클릭시 index 페이지로 이동		
			table	자동차보험 연간 지급 가능한 청구서를 그래프로 기록		
			button	'뒤로가기' 클릭시 청구서 입력 페이지로 이동		
			button	'홈으로' 클릭시 index 페이지로 이동		

13. 스토리보드

ID	account_01	화면 제목	계좌이체
화면 경로	계좌이체		
<p>1 계좌정보 입력</p> <p>2 계좌번호 <input type="text"/></p> <p>고객성함 <input type="text"/></p> <p>고객번호 <input type="text"/></p> <p>3 <input type="button" value="확인"/> <input type="button" value="뒤로가기"/></p>			

번호	상세 설명
1	페이지 헤더 영역 <ul style="list-style-type: none">페이지 타이틀
2	페이지 입력 영역 <ul style="list-style-type: none">고객의 계좌 정보를 입력
3	페이지 이동 영역 <ul style="list-style-type: none">확인 버튼 클릭시 고객계좌 상세페이지로 이동뒤로가기 버튼 클릭시 이전 페이지로 이동
관련 테이블 정의서	
ACCMEMBERS ACCOUNT TRANSFERINFO	

13. 스토리보드

ID	account_02	화면 제목	고객 계좌 상세페이지																		
화면 경로	고객 계좌 상세페이지																				
 <p>The screenshot shows a customer account detail page. At the top, there's a header with the title '고객 계좌 상세페이지'. Below it, a section titled '1 계좌결제' contains the sub-section '부당 거래 확인 결과'. A red box labeled '2' highlights a table with the following data:</p> <table border="1"><tbody><tr><td>계좌번호</td><td>1234567891</td></tr><tr><td>고객성함</td><td>고동연</td></tr><tr><td>고객 생년월일</td><td>1996-05-01</td></tr><tr><td>고객 주소</td><td>남양주</td></tr><tr><td>고객 상세주소</td><td>덕소</td></tr><tr><td>고객 우편번호</td><td>12377</td></tr><tr><td>고객 연락처</td><td>01043482565</td></tr><tr><td>계좌 개설일</td><td>2005-12-01</td></tr><tr><td>고객 다른 계좌번호</td><td></td></tr></tbody></table> <p>At the bottom of this section, a red box labeled '3' contains the text '이상 입금 발견' and two buttons: '홈으로' and '통계확인'.</p>				계좌번호	1234567891	고객성함	고동연	고객 생년월일	1996-05-01	고객 주소	남양주	고객 상세주소	덕소	고객 우편번호	12377	고객 연락처	01043482565	계좌 개설일	2005-12-01	고객 다른 계좌번호	
계좌번호	1234567891																				
고객성함	고동연																				
고객 생년월일	1996-05-01																				
고객 주소	남양주																				
고객 상세주소	덕소																				
고객 우편번호	12377																				
고객 연락처	01043482565																				
계좌 개설일	2005-12-01																				
고객 다른 계좌번호																					

번호	상세 설명
1	페이지 헤더영역 <ul style="list-style-type: none">페이지 타이틀
2	고객 상세 정보 영역 <ul style="list-style-type: none">고객의 계좌번호, 이름, 생년월일, 주소 상세주소, 우편번호, 연락처, 계좌 개설일, 다른 계좌번호를 출력
3	페이지 이동 영역 <ul style="list-style-type: none">홈으로 버튼 클릭시 메인페이지로 이동통계확인 버튼 클릭시 통계차트 확인 페이지로 이동
관련 테이블 정의서	
CUSTOMERS CARD PURCHASE	

13. 스토리보드

ID	creditCard_01	화면 제목	해외카드결제
화면 경로	해외카드결제		
<p>1 관리자 페이지로 이동</p> <p>2 결제 정보 입력</p> <p>3 결제 ID: pur100032</p> <p>고객 ID: <input type="text"/></p> <p>결제 금액: <input type="text"/></p> <p>카드번호: <input type="text"/></p> <p>카드유효기간: <input type="text"/> / <input type="text"/></p> <p>IP 주소: <input type="text"/></p> <p>4 제출</p>			

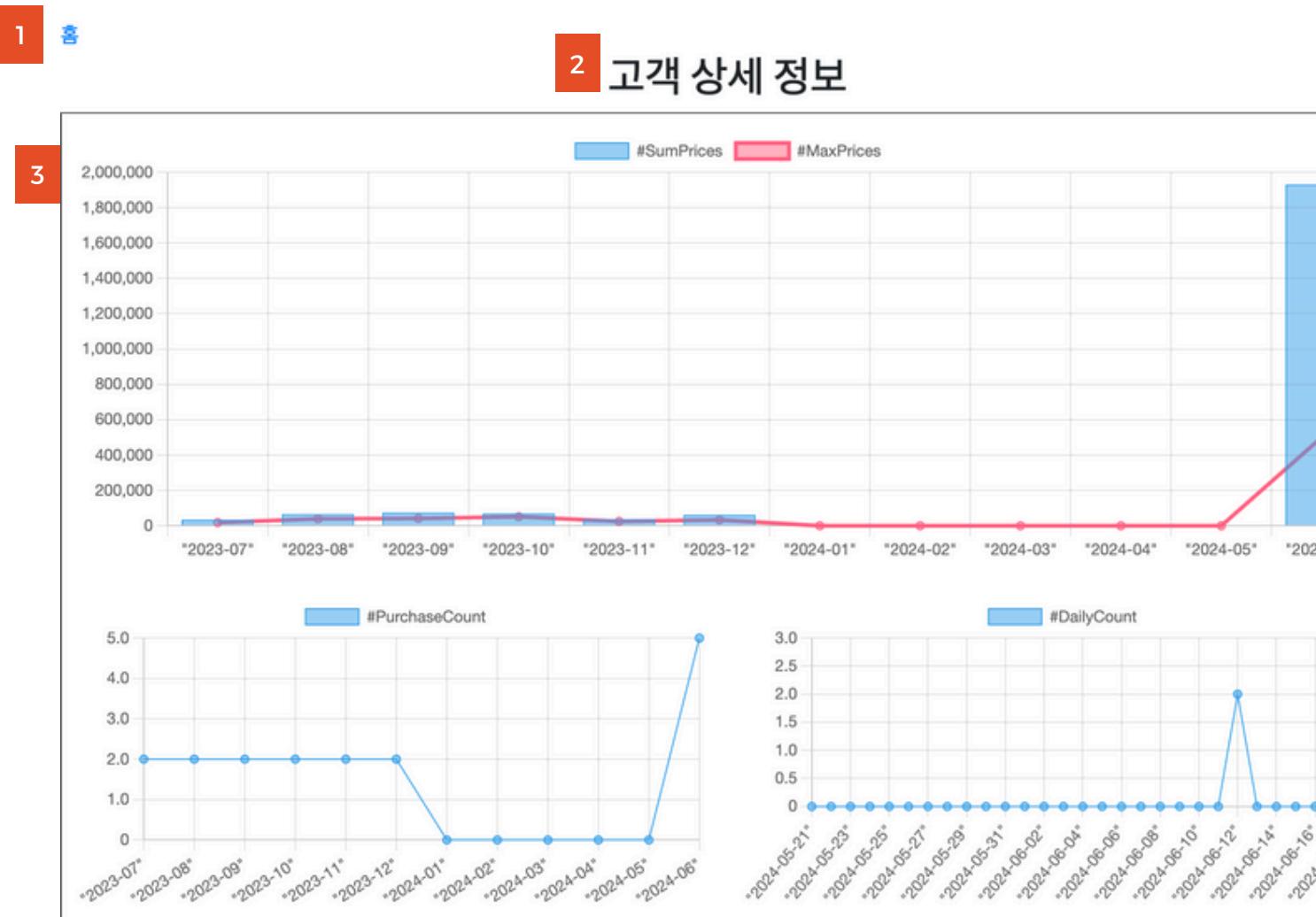
번호	상세 설명
1	Header영역 <ul style="list-style-type: none">관리자 페이지로 이동 링크
2	페이지 헤더영역 <ul style="list-style-type: none">페이지 타이틀
3	페이지 입력 영역 <ul style="list-style-type: none">결제를 하기위한 정보들을 입력
4	페이지 입력값 제출 영역 <ul style="list-style-type: none">입력된 정보들을 제출하여 FDS요인 분석 후 결과값을 출력하는 페이지로 이동
관련 테이블 정의서	
CUSTOMERS CARD PURCHASE	

13. 스토리보드

ID	creditCard_02_01	화면 제목	관리자페이지
화면 경로	관리자페이지		
1 홈	2 이상 거래 탐지 고객 리스트		
3 결제번호	카드번호	고객번호	
pur100031	crd100001	ctm100001	

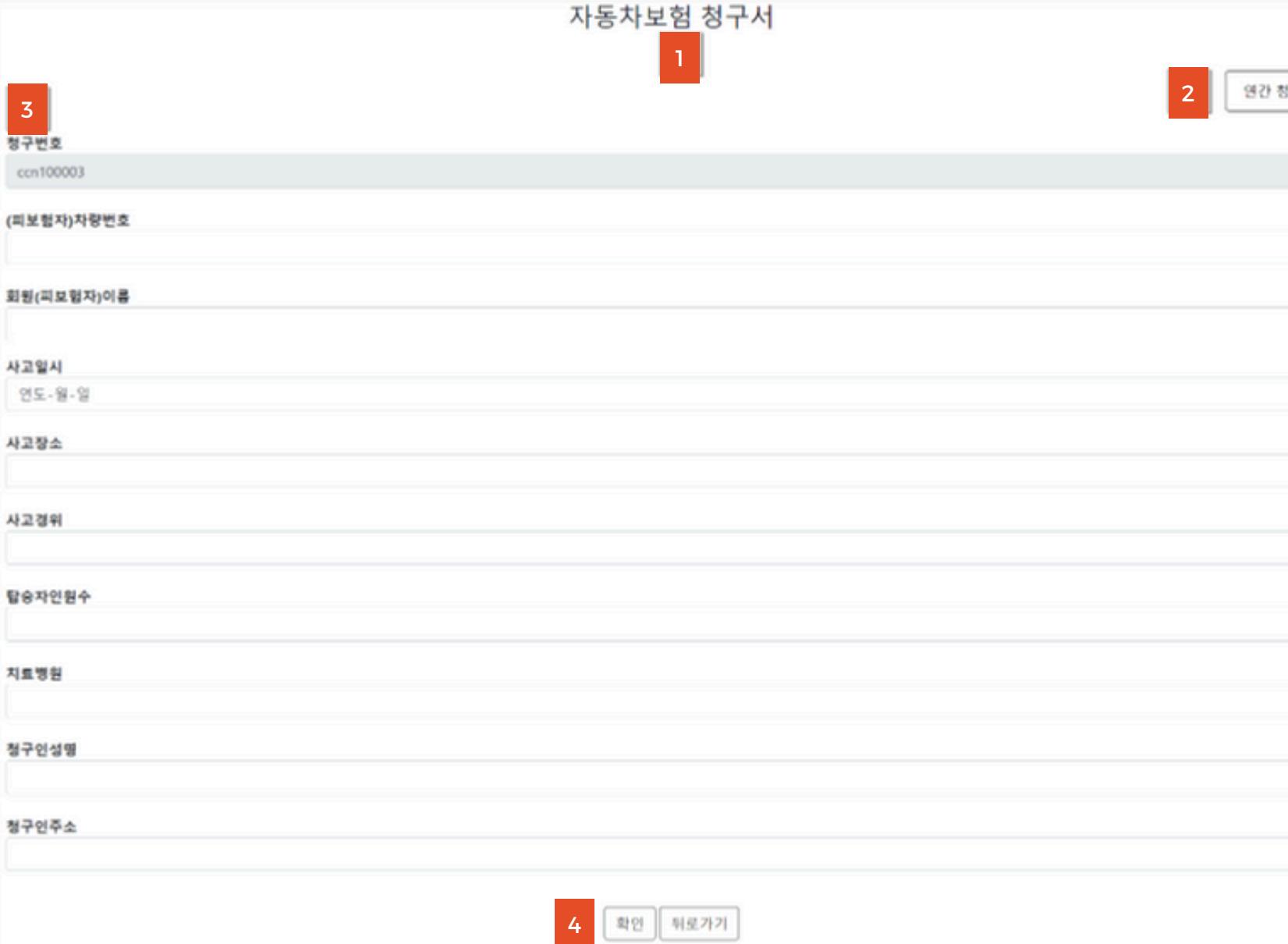
번호	상세 설명
1	Header영역 <ul style="list-style-type: none">메인페이지로 이동
2	페이지 헤더영역 <ul style="list-style-type: none">페이지 타이틀
3	고객리스트 영역 <ul style="list-style-type: none">비정상적인 거래요청을 한 고객들의 리스트를 출력
관련 테이블 정의서	
CUSTOMERS CARD PURCHASE	

13. 스토리보드

ID	creditCard_02_02	화면 제목	고객상세페이지			
화면 경로	관리자페이지 > 고객상세페이지					
 <p>The screenshot shows the 'Customer Detail Page'. At the top left is a 'Home' button (1). In the center is the title 'Customer Detail Information' (2). Below the title are three charts: 1) A bar chart showing monthly total spending (#SumPrices) from July 2023 to June 2024, with a significant spike in June. 2) A line chart showing the number of purchases per month (#PurchaseCount) from July 2023 to June 2024, with a sharp increase in June. 3) A line chart showing daily purchase counts (#DailyCount) from May 21 to June 18, 2024, showing a single peak on June 12. At the bottom is a summary table (4) with columns: Card Number, Payment Amount, Payment Date, Customer Phone Number, and Payment Status. The payment status has two buttons: 'Payment Approved' (green) and 'Payment Declined' (red).</p>						
4	카드번호 9999-8888-7777-6666	결제 금액 1500000원	결제일 2024-06-14	고객 전화번호 01012345678	결제 승인 결제 승인	결제 취소 결제 취소

번호	상세 설명
1	Header 영역 <ul style="list-style-type: none">메인페이지로 이동
2	페이지 헤더 영역 <ul style="list-style-type: none">페이지 타이틀
3	고객 결제 패턴 분석 영역 <ul style="list-style-type: none">최근 1년 매달 총사용금액, 평균사용금액 차트 출력최근 1년 매달 결제 횟수 차트 출력최근 1달동안의 결제 패턴 차트 출력
4	결제 승인/취소 영역 <ul style="list-style-type: none">차트 분석후 정상 결제면 승인 그렇지 않으면 취소
관련 테이블 정의서	
CUSTOMERS CARD PURCHASE	

13. 스토리보드

ID	carClaim_01	화면 제목	자동차 보험 청구서	번호	상세 설명
화면 경로		자동차 보험 청구서		1	페이지 헤더영역 <ul style="list-style-type: none">페이지 타이틀
				2	연간 청구수 페이지 이동 영역 <ul style="list-style-type: none">연간 청구수 버튼 클릭시 통계자료 페이지로 이동
				3	페이지 입력 영역 <ul style="list-style-type: none">청구번호는 자동 부여차량번호, 피보험자 이름, 사고일시, 사고장소, 사고 경위, 탑승자인원수, 치료병원, 청구인성명, 청구인 주소를 입력
				4	페이지 이동 영역 <ul style="list-style-type: none">확인 버튼 클릭시 청구 결제후 출력 페이지로 이동뒤로가기 버튼 클릭시 이전 페이지로 이동
					관련 테이블 정의서
					CARMEMBERS CARCLAIM

13. 스토리보드

ID	creditCard_02_02	화면 제목	부당 지급 의심 확인																											
화면 경로	부당 지급 의심 확인																													
<p>1 자동차보험 부당 지급 의심 확인 결과</p> <table><tbody><tr><td>청구번호</td><td>ccn100003</td></tr><tr><td>차량번호</td><td>123가1234</td></tr><tr><td>회원(피보험자)이름</td><td>김승한</td></tr><tr><td>회원(피보험자)차종</td><td>현대 아반떼</td></tr><tr><td>회원(피보험자)연락처</td><td>010-1234-5678</td></tr><tr><td>면허번호</td><td>12-34-567890-12</td></tr><tr><td>면허종류</td><td>1종 보통</td></tr><tr><td>사고일시</td><td>Fri Jun 28 00:00:00 KST 2024</td></tr><tr><td>사고장소</td><td>서울특별시 강남구 역삼동 근처 교차로 신호대기 중 후미 추돌한 사고</td></tr><tr><td>사고경위</td><td>교차로 신호대기 중 후미 추돌한 사고</td></tr><tr><td>탑승자인원수</td><td>4</td></tr><tr><td>치료병원</td><td>삼성서울병원</td></tr><tr><td>청구인성명</td><td>김승환</td></tr><tr><td>청구인주소</td><td>서울특별시 강남구 역삼동 주공아파트</td></tr></tbody></table> <p>2</p> <p>3 지급 보류</p> <p>4 사유: 이전 사고와 비슷한 사고경위로 인해 클레임이 보류되었습니다. 최근 6개월 내에 사고가 재발하여 클레임이 보류되었습니다.</p> <p>뒤로가기 홈으로</p>			청구번호	ccn100003	차량번호	123가1234	회원(피보험자)이름	김승한	회원(피보험자)차종	현대 아반떼	회원(피보험자)연락처	010-1234-5678	면허번호	12-34-567890-12	면허종류	1종 보통	사고일시	Fri Jun 28 00:00:00 KST 2024	사고장소	서울특별시 강남구 역삼동 근처 교차로 신호대기 중 후미 추돌한 사고	사고경위	교차로 신호대기 중 후미 추돌한 사고	탑승자인원수	4	치료병원	삼성서울병원	청구인성명	김승환	청구인주소	서울특별시 강남구 역삼동 주공아파트
청구번호	ccn100003																													
차량번호	123가1234																													
회원(피보험자)이름	김승한																													
회원(피보험자)차종	현대 아반떼																													
회원(피보험자)연락처	010-1234-5678																													
면허번호	12-34-567890-12																													
면허종류	1종 보통																													
사고일시	Fri Jun 28 00:00:00 KST 2024																													
사고장소	서울특별시 강남구 역삼동 근처 교차로 신호대기 중 후미 추돌한 사고																													
사고경위	교차로 신호대기 중 후미 추돌한 사고																													
탑승자인원수	4																													
치료병원	삼성서울병원																													
청구인성명	김승환																													
청구인주소	서울특별시 강남구 역삼동 주공아파트																													

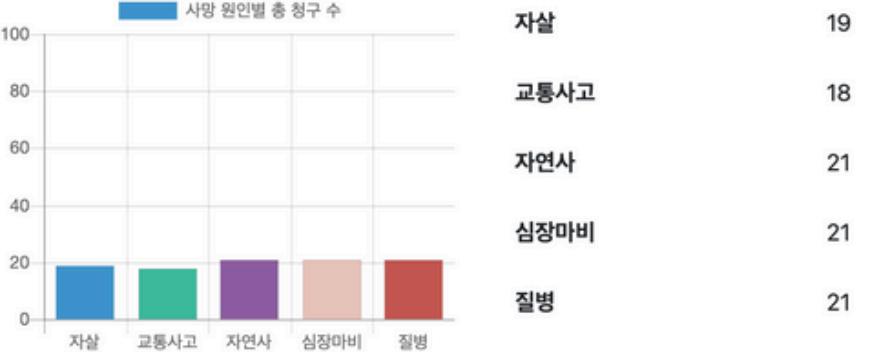
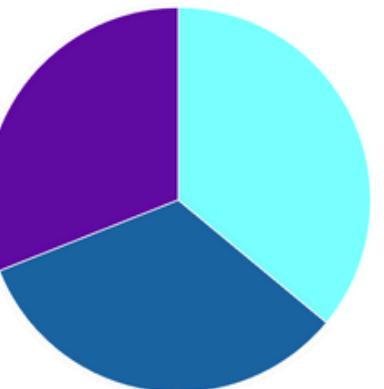
번호	상세 설명
1	페이지 헤더영역 <ul style="list-style-type: none">페이지 타이틀
2	피보험자 정보 영역 <ul style="list-style-type: none">청구번호, 차량번호, 이름, 차종, 연락처, 면허번호, 면허종류, 사고일시, 사고장소, 사고경위, 탑승자인원수, 치료병원, 청구인성명, 청구인주소 정보 출력
3	지급 상태 영역 <ul style="list-style-type: none">지급 보류 또는 지급 승인을 출력
4	지급 상태 사유 영역 <ul style="list-style-type: none">지급이 보류된 상태라면 그에 해당한 사유를 출력
관련 테이블 정의서	
CARMEMBERS CARCLAIM	

13. 스토리보드

ID	creditCard_02_02	화면 제목	연간 청구수
화면 경로	연간 청구수		
			

번호	상세 설명
1	페이지 헤더 영역 <ul style="list-style-type: none">페이지 타이틀
2	통계 차트 영역 <ul style="list-style-type: none">1년간의 연간 청구수를 차트로 출력
3	페이지 이동 영역 <ul style="list-style-type: none">뒤로가기 버튼 클릭시 이전 페이지로 이동홈으로 버튼 클릭시 메인페이지로 이동
관련 테이블 정의서	
CARMEMBERS CARCLAIM	

13. 스토리보드

ID	deathClaim_01	화면 제목	사망보험통계분석	번호	상세 설명																						
화면 경로	사망보험통계분석			1	Header영역 <ul style="list-style-type: none"> FDSProjectTeam 클릭시 메인페이지로 이동 현재 로그인된 사람 현재 청구 수 																						
	<p>1 FDSProjectTeam Home About Claims 관리자님 환영합니다. 현재 청구 수 총 : 100개 입니다.</p> <p>2 사망보험 청구 통계 및 분석</p> <p>3 청구서 목록 부당 지급 확인 지금 거부 통계 리포트</p>  <table border="1"> <thead> <tr> <th>사망 원인별 총 청구 수</th> <th>자살</th> <th>19</th> </tr> </thead> <tbody> <tr> <td></td> <td>교통사고</td> <td>18</td> </tr> <tr> <td></td> <td>자연사</td> <td>21</td> </tr> <tr> <td></td> <td>심장마비</td> <td>21</td> </tr> <tr> <td></td> <td>질병</td> <td>21</td> </tr> </tbody> </table> <p>사망 장소</p>  <table border="1"> <thead> <tr> <th>사망 장소</th> <th>병원</th> <th>36</th> </tr> </thead> <tbody> <tr> <td></td> <td>자택</td> <td>33</td> </tr> <tr> <td></td> <td>기타</td> <td>31</td> </tr> </tbody> </table> <p>리포트</p> <p>사망보험 청구 추이</p> <p>COVID-19 감염병 질환 사망으로 인한 사망보험 청구 [위험] 질병 사망 증가로 인해 청구 증가 [주의] 자연사 사망 증가로 인해 청구 증가</p> <p>피보험자와 실제 청구인의 관계가 불확실한 청구</p>	사망 원인별 총 청구 수	자살	19		교통사고	18		자연사	21		심장마비	21		질병	21	사망 장소	병원	36		자택	33		기타	31	2	사망보험 청구 통계 영역 <ul style="list-style-type: none"> 사망 원인별 총 청구수에 관한 통계 자료 사망 장소에 관한 통계 자료
사망 원인별 총 청구 수	자살	19																									
	교통사고	18																									
	자연사	21																									
	심장마비	21																									
	질병	21																									
사망 장소	병원	36																									
	자택	33																									
	기타	31																									
		3	관리자 메뉴 영역 <ul style="list-style-type: none"> 청구서 목록 링크 클릭시 청구서 리스트로 이동 부당 지급 확인 링크 클릭시 청구서 부당지급확인페이지로 이동 																								
					관련 테이블 정의서																						
					DEATH CLAIM INSURED																						

13. 스토리보드

ID	화면 제목	관리자페이지
화면 경로	관리자페이지	

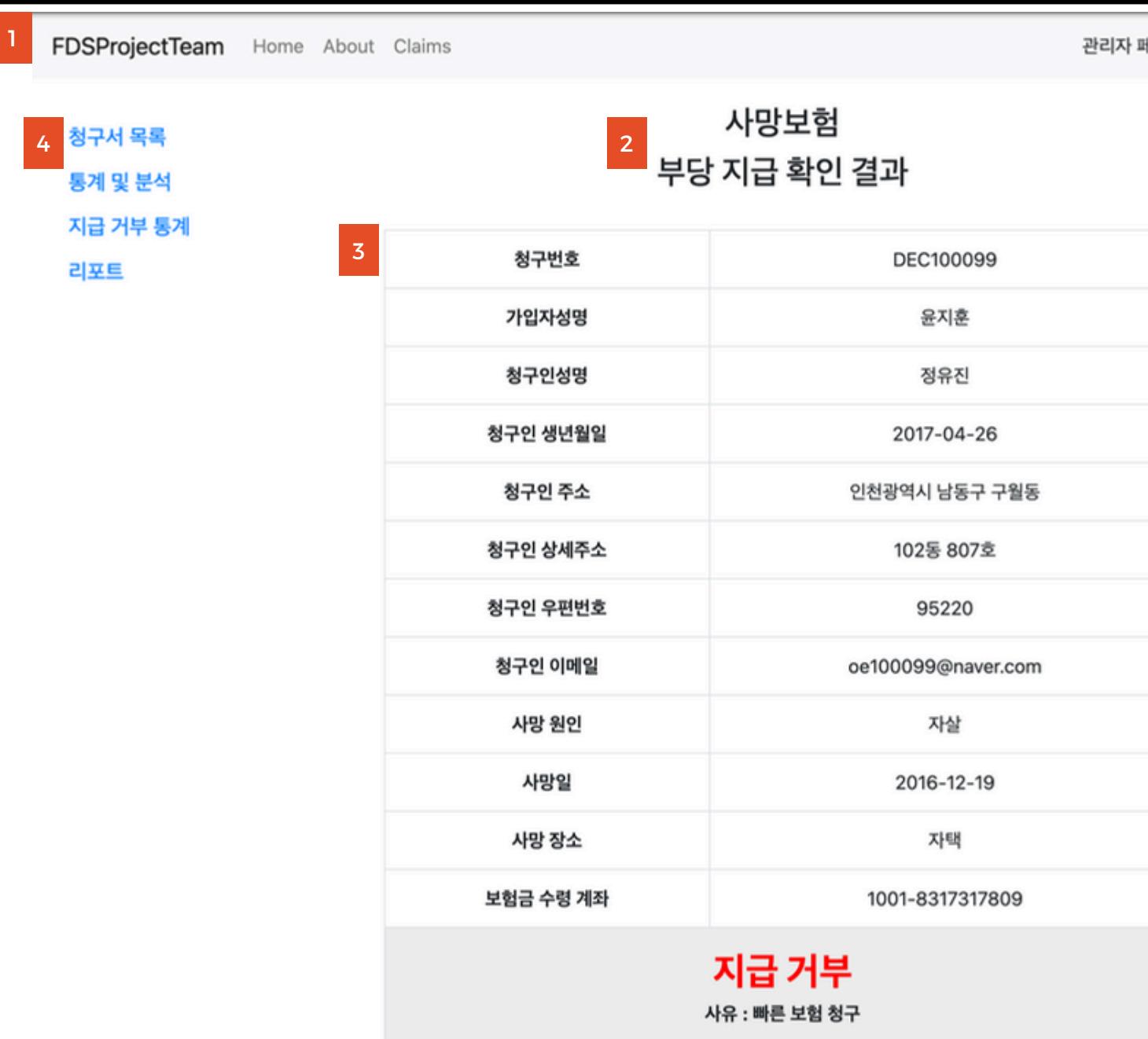
번호	상세 설명
1	Header 영역 <ul style="list-style-type: none"> FDSProjectTeam 클릭시 메인페이지로 이동
2	청구 목록 영역 <ul style="list-style-type: none"> 등록된 청구서 리스트 출력
3	관리자 메뉴 영역 <ul style="list-style-type: none"> 청구서 목록 링크 클릭시 청구서 리스트로 이동 부당 지급 확인 링크 클릭시 청구서 부당지급확인페이지로 이동
관련 테이블 정의서	
DEATH CLAIM INSURED	

13. 스토리보드

ID	deathClaim_03_01	화면 제목	부당 지급 확인 입력
화면 경로	부당 지급 확인 입력		
<p>1 FDSProjectTeam Home About Claims 관리자 페이지</p> <p>2 청구서 가져오기</p> <p>3 확인 뒤로가기</p> <p>4 청구서 목록 통계 및 분석 지급 거부 통계 리포트</p>			

번호	상세 설명
1	Header 영역 <ul style="list-style-type: none"> FDSProjectTeam 클릭시 메인페이지로 이동
2	정보 입력 영역 <ul style="list-style-type: none"> 청구서 버튼 클릭시 청구서 리스트에서 정보들을 가져올 수 있음 청구서 정보들을 입력
3	페이지 이동 영역 <ul style="list-style-type: none"> 확인 버튼 클릭시 부당 지급확인 출력 페이지로 이동 뒤로가기 버튼 클릭시 이전페이지로 이동
4	관리자 메뉴 영역 <ul style="list-style-type: none"> 청구서 목록 링크 클릭시 청구서 리스트로 이동 통계 및 분석 링크 클릭시 통계 페이지로 이동
관련 테이블 정의서	
DEATH CLAIM INSURED	

13. 스토리보드

ID	deathClaim_03_02	화면 제목	부당지급확인결과																							
화면 경로	부당지급확인결과																									
 <p>The screenshot displays a web application interface for a death claim. At the top left is the navigation bar: 'FDSProjectTeam' (highlighted with a red box), 'Home', 'About', 'Claims', and '관리자 페이지' (highlighted with a grey box). On the left side, there's a sidebar with four items: '청구서 목록' (highlighted with a red box), '통계 및 분석', '지급 거부 통계', and '리포트'. In the center, the main content area has a title '사망보험' and '부당 지급 확인 결과'. Below this is a table with the following data:</p> <table border="1"> <tbody> <tr> <td>청구번호</td> <td>DEC100099</td> </tr> <tr> <td>가입자성명</td> <td>윤지훈</td> </tr> <tr> <td>청구인성명</td> <td>정유진</td> </tr> <tr> <td>청구인 생년월일</td> <td>2017-04-26</td> </tr> <tr> <td>청구인 주소</td> <td>인천광역시 남동구 구월동</td> </tr> <tr> <td>청구인 상세주소</td> <td>102동 807호</td> </tr> <tr> <td>청구인 우편번호</td> <td>95220</td> </tr> <tr> <td>청구인 이메일</td> <td>oe100099@naver.com</td> </tr> <tr> <td>사망 원인</td> <td>자살</td> </tr> <tr> <td>사망일</td> <td>2016-12-19</td> </tr> <tr> <td>사망 장소</td> <td>자택</td> </tr> <tr> <td>보험금 수령 계좌</td> <td>1001-8317317809</td> </tr> </tbody> </table> <p>At the bottom of the main content area, there's a red button labeled '지급 거부' (highlighted with a red box) and the text '사유 : 빠른 보험 청구'.</p>			청구번호	DEC100099	가입자성명	윤지훈	청구인성명	정유진	청구인 생년월일	2017-04-26	청구인 주소	인천광역시 남동구 구월동	청구인 상세주소	102동 807호	청구인 우편번호	95220	청구인 이메일	oe100099@naver.com	사망 원인	자살	사망일	2016-12-19	사망 장소	자택	보험금 수령 계좌	1001-8317317809
청구번호	DEC100099																									
가입자성명	윤지훈																									
청구인성명	정유진																									
청구인 생년월일	2017-04-26																									
청구인 주소	인천광역시 남동구 구월동																									
청구인 상세주소	102동 807호																									
청구인 우편번호	95220																									
청구인 이메일	oe100099@naver.com																									
사망 원인	자살																									
사망일	2016-12-19																									
사망 장소	자택																									
보험금 수령 계좌	1001-8317317809																									

번호	상세 설명
1	Header 영역 <ul style="list-style-type: none"> FDSProjectTeam 클릭시 메인페이지로 이동
2	페이지 헤더 영역 <ul style="list-style-type: none"> 페이지 타이틀
3	사망보험 결과 확인 영역 <ul style="list-style-type: none"> 청구서에 관한 정보들 출력 지급 완료 혹은 지급 거부 출력 지급 거부시 사유 출력
4	관리자 메뉴 영역 <ul style="list-style-type: none"> 청구서 목록 링크 클릭시 청구서 리스트로 이동 통계 및 분석 링크 클릭시 통계 페이지로 이동
관련 테이블 정의서	
DEATH CLAIM INSURED	

14. 소스코드

계좌이체

VIEW

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" type="text/css" href="../static/bootstrap/bootstrap.min.css">
</head>
<body>
<h3 style="text-align: center">계좌정보 입력</h3>
<form action="accountCheck" method="post">
<table class="table table-borderless">
<tr><th>계좌번호<br /><input type="text" class="form-control" name="accountNum" required="required"/></th></tr>
<tr><th>고객성함<br /><input type="text" class="form-control" name="accmemName" required="required"/></th></tr>
<tr><th>고객번호<br /><input type="text" class="form-control" name="accmemNum" required="required"/></th></tr>
</table>
<table class="table table-borderless" style="text-align: center;">
<tr><th>
<input type="submit" class="btn btn-outline-dark" value="확인">
<button type="button" class="btn btn-outline-dark" onclick="javascript:history.back()>뒤로가기</button>
</th></tr>
</table>
</form>
</body>
</html>
```

CONTROLLER

```
@Controller
public class AccountController {
    @Autowired
    AccountResultService accountResultService;
    @Autowired
    AccountChartService accountChartService;

    @GetMapping("accountCheck")
    public String accountCheck(@RequestParam("accountNum") String accountNum,
                               @RequestParam("accmemName") String accmemName,
                               @RequestParam("accmemNum") String accmemNum) {
        accountResultService.execute(model, accountNum, accmemName, accmemNum);
        return "thymeleaf/account/accountResult";
    }
}
```

SERVICE

```
public class AccountResultService {
    @Autowired
    AccountMapper accountMapper;
    public void execute(Model model, String accountNum, String accmemName, String accmemNum) {
        AccountMemberTO amto = accountMapper.accountMemberSelect(accountNum, accmemName, accmemNum);
        model.addAttribute("amto", amto);
        List list = new ArrayList<>();
        list = accountMapper.transferInfoList(accountNum);

        Integer incomeTotal = 0;
        Integer outcomeTotal = 0;
        Integer incomeCount = 0;
        Integer outcomeCount = 0;
        for(AccTransferInfoTO dto : list) {
            if(dto.getIncomePrice() > 0) {
                incomeTotal += dto.getIncomePrice();
                incomeCount++;
            }
            if(dto.getOutcomePrice() > 0) {
                outcomeTotal += dto.getOutcomePrice();
                outcomeCount++;
            }
        }
        Integer avgIncome = 0;
        Integer avgOutcome = 0;
        if(incomeCount > 10) {
            avgIncome = incomeTotal / incomeCount;
        }
        if(outcomeCount > 10) {
            avgOutcome = outcomeTotal / outcomeCount;
        }

        boolean avgIncomDifference = false;
        boolean avgOutcomDifference = false;
        boolean sequenceFound = false;
        boolean cashoutAbnormal = false;
        SimpleDateFormat sdf = new SimpleDateFormat(pattern: "yyyy-MM-dd");
        String previousDate = "";
        Integer dateCount = 0;
        Integer outcomeDateCount = 0;
        String cashoutDate = "";
        for(AccTransferInfoTO dto : list) {
            //1. 평균금액
            if(dto.getIncomePrice() > avgIncome*200) {
                avgIncomDifference = true;
            }
            if(dto.getOutcomePrice() > avgOutcome*200) {
                avgOutcomDifference = true;
            }

            //2. 같은날 큰 금액이 여러번 발생했을때
            String incomeDate = sdf.format(dto.getIncomeDate());
            String outcomeDate = sdf.format(dto.getOutcomeDate());
            //income
            if(dto.getIncomePrice() > avgIncome) {
                if(incomeDate.equals(previousDate)) {
                    dateCount++;
                    if(dateCount > 5) {
                        sequenceFound = true;
                    }
                } else {
                    previousDate = incomeDate;
                    dateCount = 1;
                }
            } else {
                dateCount = 0;
            }
            //outcome
            if(dto.getOutcomePrice() > avgOutcome) {
                if(outcomeDate.equals(previousDate)) {
                    dateCount++;
                    if(dateCount > 5) {
                        sequenceFound = true;
                    }
                } else {
                    previousDate = outcomeDate;
                    dateCount = 1;
                }
            }
        }

        //3. 현금인출이 많을때
        //현금인출이 200이상의 같은날 2번이상 인출할때
        outcomeDate = sdf.format('2000000 & dto.getTransferContent());
        if(dto.getOutcomePrice() > 2000000 & dto.getTransferContent().equals("현금인출")) {
            if(outcomeDate.equals(cashoutDate)) {
                outcomeDateCount++;
                if(outcomeDateCount >= 2) {
                    cashoutAbnormal = true;
                }
            } else {
                cashoutDate = outcomeDate;
                outcomeDateCount = 1;
            }
        } else {
            outcomeDateCount = 0;
        }
    }
    model.addAttribute("avgIncomDifference", avgIncomDifference);
    model.addAttribute("avgOutcomDifference", avgOutcomDifference);
    model.addAttribute("sequenceFound", sequenceFound);
    model.addAttribute("cashoutAbnormal", cashoutAbnormal);
}
```

REPOSITORY

```
public interface AccountMapper {
    public AccountMemberDTO accountMemberSelect(String accountNum, String accmemName, String accmemNum);
    public List<AccTransferInfoTO> transferInfoList(String accountNum);
}

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC
    "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="fdsprojectteam.mapper.AccountMapper">
    <resultMap id="accountResultMap" type="account">
        <constructor>
            <idArg column="ACCOUNT_NUM" javaType="string" jdbcType="VARCHAR" />
            <arg column="ACCMEM_NUM" javaType="string" jdbcType="VARCHAR" />
            <arg column="ACCOUNT_DATE" javaType="java.util.Date" jdbcType="DATE" />
        </constructor>
    </resultMap>

    <resultMap id="accmemberResultMap" type="accmember">
        <constructor>
            <idArg column="ACCMEM_NUM" javaType="string" jdbcType="VARCHAR" />
            <arg column="ACCMEM_NAME" javaType="string" jdbcType="VARCHAR" />
            <arg column="ACCMEM_ADDR" javaType="string" jdbcType="VARCHAR" />
            <arg column="ACCMEM_DETAIL" javaType="string" jdbcType="VARCHAR" />
            <arg column="ACCMEM_POST" javaType="string" jdbcType="VARCHAR" />
            <arg column="ACCMEM_TEL" javaType="string" jdbcType="VARCHAR" />
            <arg column="ACCMEM_BIRTH" javaType="java.util.Date" jdbcType="DATE" />
        </constructor>
    </resultMap>

    <resultMap id="transferInfoResultMap" type="transferInfo">
        <id column="TRANSFER_NUM" javaType="string" jdbcType="VARCHAR" property="transferNum"/>
        <result column="ACCMEM_NUM" javaType="string" jdbcType="VARCHAR" property="accmemNum"/>
        <result column="ACCOUNT_NUM" javaType="string" jdbcType="VARCHAR" property="accountNum"/>
        <result column="INCOME_PRICE" javaType="integer" jdbcType="BIGINT" property="incomePrice"/>
        <result column="INCOME_DATE" javaType="java.util.Date" jdbcType="DATE" property="incomeDate"/>
        <result column="OUTCOME_PRICE" javaType="integer" jdbcType="BIGINT" property="outcomePrice"/>
        <result column="OUTCOME_DATE" javaType="java.util.Date" jdbcType="DATE" property="outcomeDate"/>
        <result column="TRANSFER_CONTENT" javaType="string" jdbcType="VARCHAR" property="transferContent"/>
    </resultMap>

    <resultMap id="accountMemberResultMap" type="accountMember">
        <association property="accmemberDTO" javaType="accmember" resultMap="accmemberResultMap"/>
        <association property="accountDTO" javaType="account" resultMap="accountResultMap"/>
    </resultMap>

    <select id="accountMemberSelect" resultMap="accountMemberResultMap" parameterType="map">
        select m.ACCEM_NUM, ACCEM_NAME, ACCEM_ADDR, ACCEM_DETAIL, ACCEM_POST, ACCEM_TEL, ACCEM_BIRTH,
               ACCOUNT_NUM, ACCOUNT_DATE
        from accmembers m join account a
        on m.acmem_num = a.acmem_num
        where ACCOUNT_NUM = #{accountNum} and ACCEM_NAME = #{accmemName} and m.ACCEM_NUM = #{accmemNum}
    </select>

    <select id="transferInfoList" resultMap="transferInfoResultMap" parameterType="string">
        select TRANSFER_NUM , ACCEM_NUM, ACCOUNT_NUM, nvl(INCOME_PRICE,0) income_Price, DongYeonKo, 6/14/24, 5:34 오후
               nvl(INCOME_DATE, TO_DATE('9999-01-01', 'YYYY-MM-DD')) income_date, nvl(OUTCOME_PRICE,0) outcome_Price,
               nvl(OUTCOME_DATE, TO_DATE('9999-01-01', 'YYYY-MM-DD')) outcome_date, TRANSFER_CONTENT
        from transferInfo
        where ACCOUNT_NUM = #{accountNum}
    </select>
</mapper>
```

14. 소스코드

해외카드결제

VIEW

```
<body>
<a href="/pgAdmin">관리자 페이지로 이동</a>
<h2 style="...">결제 정보 입력</h2>
<form action="/payment" method="post">
<table class="table table-borderless" >
<tr>
<th>
    결제 ID:<br>
    <input type="text" class="form-control" name="purchaseId" th:value="${purchaseCommand.purchaseId}" required>
</th>
</tr>
<tr> You, 6/17/24, 4:32 오후 · Uncommitted changes
<th>
    고객 ID:<br>
    <input type="text" class="form-control" name="customerId" required>
</th>
</tr>
<tr>
<th>
    결제 금액:<br>
    <input type="number" class="form-control" name="purchasePrice" required>
</th>
</tr>
<tr>
<th>
    카드번호:<br>
    <input type="text" class="form-control" name="cardNum" required>
</th>
</tr>

```

CONTROLLER

```
@Controller
@RequiredArgsConstructor
public class CreditCardController {

    private final PurchaseAutoNumSelectService purchaseAutoNumSelectService;
    private final PaymentService paymentService;
    private final IPAddressService ipAddressService;
    private final IPBlockService ipBlockService;
    private final PurchaseErrorListService purchaseErrorListService;
    private final CustomerDetailService customerDetailService;
    private final ApprovePaymentService approvePaymentService;
    private final CancelPaymentService cancelPaymentService;

    @GetMapping("creditCard")
    public String creditCard(Model model){
        purchaseAutoNumSelectService.execute(model);
        return "thymeleaf/purchase/creditCardForm";
    }

    @PostMapping("payment")
    public String payment(Model model, PurchaseCommand purchaseCommand){
        // request.getRemoteAddr() : 입력한 사람의 IP주소를 받아오는 명령어
        int i = paymentService.execute(purchaseCommand, model);
        // System.out.println(i);
        if (i == 0) {
            return "thymeleaf/purchase/creditCardSuccess";
        }else{
            model.addAttribute( attributeName: "errorCode", i);
            return "thymeleaf/purchase/creditCardFail";
        }
    }
}
```

MAPPER

```
<select id="cardSelectOne" resultMap="cardResultMap" parameterType="string">
    select <include refid="cardBaseColumns" /> wndudtjd, 6/10/24, 3:08 오후 · 중간커밋
    from card
    where CARD_NUM = #{cardNum}
</select>
<select id="countrySelectOne" resultMap="countryResultMap" parameterType="long">
    select <include refid="countryBaseColumns" />
    from country
    where #{ipAddress} between IP_START_NUM and IP_END_NUM
</select>
<update id="errorCountUpdate" parameterType="hashmap">
    update card
    set ERROR_COUNT = #{errorCount}
    where CARD_NUM = #{cardNum}
</update>
<update id="tradingHaltUpdate" parameterType="card">
    merge into TRADING_HALT t
    using (select CARD_ID from CARD where CARD_ID = #{cardId}) c
    on (c.CARD_ID = t.CARD_ID)
    when matched then
        update set HALT_DATE = null
        delete where CARD_ID = #{cardId}
    when not matched then
        insert (<include refid="tradingHaltBaseColumns" />
        values (#{cardId}, #{customerId}, sysdate)
</update>
<select id="tradingHaltSelectOne" parameterType="string" resultMap="tradingHaltResultMap">
    select <include refid="tradingHaltBaseColumns" />
    from TRADING_HALT
    where CARD_ID = #{cardId}
</select>
<select id="avgPriceSelect" parameterType="string" resultType="integer">
    <![CDATA[
        SELECT NVL(ROUND(AVG(PURCHASE_PRICE), 0), 0)
        FROM purchase
        WHERE CARD_ID = #{cardId}
        AND TRUNC(PURCHASE_DATE) >= TRUNC(ADD_MONTHS(SYSDATE, -1))
    ]]>

```

REPOSITORY

```
@Mapper 15 usages wndudtjd
public interface PaymentMapper {
    CardDTO cardSelectOne(String cardNum); 1 usage wndudtjd
    CountryDTO countrySelectOne(Long ipAddress); 1 usage wndudtjd
    void errorCountUpdate(Map<String, Object> map); 1 usage wndudtjd
    void tradingHaltUpdate(CardDTO cardDTO); 1 usage wndudtjd
    TradingHaltDTO tradingHaltSelectOne(String cardNum); 1 usage wndudtjd
    int avgPriceSelect(String cardId); 1 usage wndudtjd
    void purchaseInsert(PurchaseDTO dto); 1 usage wndudtjd
    int purchaseCount(String customerId); 1 usage wndudtjd
    void cardErrorCountUpdate(String cardId); 2 usages wndudtjd
    void ipAgreeInsert(Map<String, Object> map); 1 usage wndudtjd
    void ipBlockInsert(Map<String, Object> map); 1 usage wndudtjd
    IPBlockDTO ipBlockSelect(Map<String, Object> map); 1 usage wndudtjd
    int ipBlockCount(long ipAddress); 1 usage wndudtjd
    IPAgreeDTO ipAgreeSelect(Map<String, Object> map); 1 usage wndudtjd
}
```

SERVICE

```
@Service 1 usage wndudtjd
@RequiredArgsConstructor
public class PaymentService {
    private final PaymentMapper paymentMapper;
    public int execute(PurchaseCommand purchaseCommand, Model model) { wndudtjd, 6/10/24, 3:08 오후 · 중간커밋
        // 입력된 IP주소를 정규화
        CardDTO cardDTO = paymentMapper.cardSelectOne(purchaseCommand.getCardNum());
        // 입력된 IP주소를 정수로
        String[] octets = purchaseCommand.getIpAddress().split(regex: "\\."); // IPv4 주소를 육진으로 분리
        long result = 0;
        for(int i = 0; i < octets.length; i++){
            result |= (Long.parseLong(octets[i]) << (24 - (i * 8))); // 육진을 변환하여 결합
        }
        // System.out.println(result);
        CountryDTO countryDTO = paymentMapper.countrySelectOne(result);
        // System.out.println(cardDTO.getCardId());
        // System.out.println(countryDTO.getCountryCode());
        int answer;
        /*
        answer = 0 => 결제 성공
        answer = 1 => 카드정지 페이징
        answer = 2 => 이용기제 페이징
        */
        if (countryDTO != null && cardDTO != null) {
            Map<String, Object> map = new HashMap<>();
            map.put("ipAddress", result);
            map.put("customerId", purchaseCommand.getCustomerId());
            IPBlockDTO ipBlockDTO = paymentMapper.ipBlockSelect(map);
            int ipBlockCount = paymentMapper.ipBlockCount(result);
            if(cardDTO.getErrorCount() <= 5 && (ipBlockDTO == null || ipBlockCount <= 5)){
                int avgPrice = paymentMapper.avgPriceSelect(cardDTO.getCardId());
                int purchaseCount = paymentMapper.purchaseCount(purchaseCommand.getCustomerId());
                IPAgreeDTO ipAgreeDTO = paymentMapper.ipAgreeSelect(map);
                PurchaseDTO dto = new PurchaseDTO();
                if(!Objects.equals(countryDTO.getCountryCode(), cardDTO.getIssueCountry()) && ipAgreeDTO == null) // 고객의 결제 지역과 해당 카드의 발행 국가를 비교하여 불일치가 발생할 경우
                    updateErrorCount(cardDTO, model);

                model.addAttribute( attributeName: "purchaseCommand", purchaseCommand);
                model.addAttribute( attributeName: "errorMessage", attributeValue: "다른 국가에서 결제 시도가 있었습니다.");
                answer = 2;
            }else if((avgPrice < 1000000 && purchaseCommand.getPurchasePrice() >= 1000000) ||
                (avgPrice >= 1000000 && (avgPrice * 2 <= purchaseCommand.getPurchasePrice() && purchaseCommand.getPurchasePrice() <= 10000000)) {
                purchaseInsert(purchaseCommand, cardDTO, countryDTO, dto, purchaseStatus: "결제보류");
                model.addAttribute( attributeName: "errorMessage", attributeValue: "비정상적인 거래금액입니다.");
                answer = 2;
            }else if(purchaseCount > 10) {
                updateErrorCount(cardDTO, model);
                model.addAttribute( attributeName: "errorMessage", attributeValue: "비정상적인 거래 횟수입니다.");
                answer = 2;
            }else // 기제 성공
                purchaseInsert(purchaseCommand, cardDTO, countryDTO, dto, purchaseStatus: "결제완료");
                paymentMapper.cardErrorCountUpdate(cardDTO.getCardId());
                answer = 0;
            }
        }else { // 이상 거래 5회 초과시, 차단된 IP이거나 다른 5명 이상의 사람이 차단한 IP일 때
            paymentMapper.tradingHaltUpdate(cardDTO);

            model.addAttribute( attributeName: "errorMessage", attributeValue: "이상 거래 5회 초과로 카드 정지 되었습니다.");
            model.addAttribute( attributeName: "errorCount", cardDTO.getErrorCount());
            answer = 2;
        }
    }else // IP정지가 잘못되었을 때, 또는 거래 정지 카드일 경우.
    TradingHaltDTO tradingHaltDTO = paymentMapper.tradingHaltSelectOne(purchaseCommand.getCardNum());
    if(tradingHaltDTO == null) {
        model.addAttribute( attributeName: "errorMessage", attributeValue: "거래 정지 카드입니다.");
        model.addAttribute( attributeName: "tradingHaltCommand", tradingHaltDTO);
    }else {
        model.addAttribute( attributeName: "errorMessage", attributeValue: "잘못된 IP로 접근하였습니다.");
    }
    answer = 1;
}
return answer;
}

private void updateErrorCount(CardDTO cardDTO, Model model) { 3 usages wndudtjd
    int newErrorCount = cardDTO.getErrorCount() + 1;
    System.out.println(newErrorCount);
    model.addAttribute( attributeName: "errorCount", newErrorCount);
    Map<String, Object> map = new HashMap<>();
    map.put("errorCount", newErrorCount);
    map.put("cardNum", cardDTO.getCardNum());
    paymentMapper.errorCountUpdate(map);
}

private void purchaseInsert(PurchaseCommand purchaseCommand, CardDTO cardDTO, CountryDTO countryDTO, PurchaseDTO dto, String purchaseStatus) { 2 usages
    BeanUtils.copyProperties(purchaseCommand, dto);
    dto.setCardId(cardDTO.getCardId());
    dto.setCountryId(countryDTO.getCountryId());
    dto.setPurchaseStatus(purchaseStatus);
    paymentMapper.purchaseInsert(dto);
}
```

14. 소스코드

관리자페이지

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" type="text/css" href="../static/bootstrap/bootstrap.min.css">
</head>
<body>
    <a href="/">You, Today · Uncommitted changes</a>
    <h2 style="text-align: center;">이상 거래 탐지 고객 리스트</h2>
    <table border="1" align="center" class="table">
        <thead>
            <tr>
                <th scope="col">결제번호</th>
                <th scope="col">카드번호</th>
                <th scope="col">고객번호</th>
            </tr>
        </thead>
        <tr th:each="dto, idx : ${list}">
            <td><a th:href="/customerPage?purchaseId=${dto.purchaseId}&cardId=${dto.cardId}&customerId=${dto.customerId}!>[${${dto.purchaseId}}]</a></td>
            <td><a th:href="/customerPage?purchaseId=${dto.purchaseId}&cardId=${dto.cardId}&customerId=${dto.customerId}!>[${${dto.cardId}}]</a></td>
            <td><a th:href="/customerPage?purchaseId=${dto.purchaseId}&cardId=${dto.cardId}&customerId=${dto.customerId}!>[${${dto.customerId}}]</a></td>
        </tr>
    </table>
</body>
</html>

@GetMapping("pgAdmin") @wndudtjd
public String pgAdmin(Model model){
    purchaseErrorListService.execute(model);    wndudtjd, 6/10/24, 3:08 오후 · 중간
    return "thymeleaf/purchase/pgAdmin";
}
```

CONTROLLER

VIEW

```
@Service 1 usage @wndudtjd
@RequiredArgsConstructor
public class PurchaseErrorListService {

    private final PaymentMapper paymentMapper;

    public void execute(Model model) { @wndudtjd
        List<PurchaseDTO> list = paymentMapper.purchaseErrorSelect();
        model.addAttribute( attributeName: "list", list);
    }
}

<select id="purchaseErrorSelect" resultMap="purchaseResultMap">
    select <include refid="purchaseBaseColumns" />
    from purchase
    where PURCHASE_STATUS = '결제보류'
</select>

List<PurchaseDTO> |purchaseErrorSelect(); 1 us@wndudtjd
```

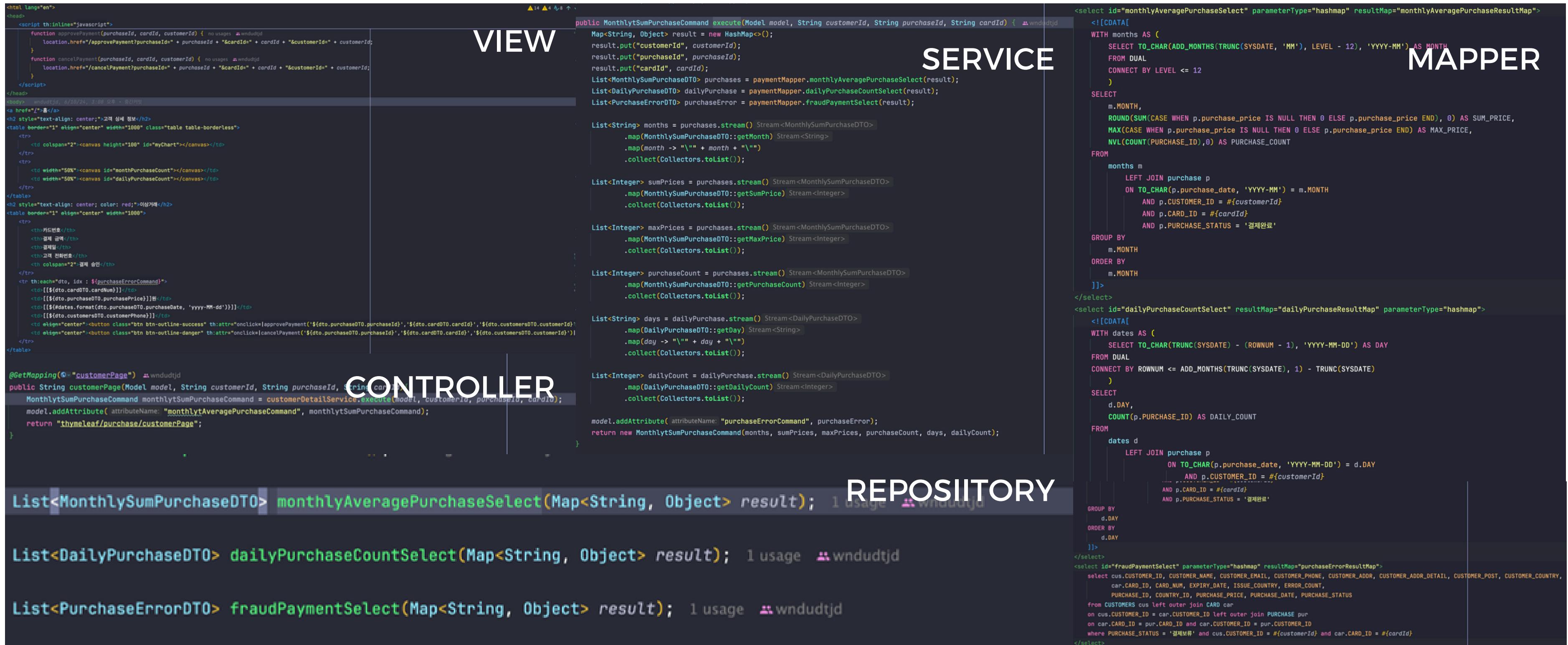
SERVICE

MAPPER

REPOSITORY

14. 소스코드

고객상세페이지



14. 소스코드

사망보험

Mapper

```
package fdsprojectteam.mapper;

import fdsprojectteam.domain.DeathClaimCountDTO;
import fdsprojectteam.domain.DeathClaimDTO;
import fdsprojectteam.domain.DeathClaimPlaceCountDTO;
import fdsprojectteam.domain.DeathClaimSearchAndPageDTO;
import org.apache.ibatis.annotations.Param;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Map;

@Repository
public interface DeathClaimMapper {
    List<DeathClaimDTO> allSelect(DeathClaimSearchAndPageDTO sepDTO);
    int deathClaimCount(String searchWord);
    List<DeathClaimCountDTO> selectCount();
    List<DeathClaimPlaceCountDTO> selectPlaceCount();
    DeathClaimDTO selectionOne(String claimNum);
}
```

SERVICE

```
package fdsprojectteam.service;

import fdsprojectteam.domain.DeathClaimCountDTO;
import fdsprojectteam.domain.DeathClaimDTO;
import fdsprojectteam.domain.DeathClaimPlaceCountDTO;
import fdsprojectteam.domain.DeathClaimSearchAndPageDTO;
import org.apache.ibatis.annotations.Param;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Map;

@Service
public class DeathClaimService {
    private final DeathClaimMapper deathClaimMapper;
    private final DeathClaimCountService deathClaimCountService;
    private final DeathClaimPlaceCountService deathClaimPlaceCountService;
    private final DeathClaimAnalyzeService deathClaimAnalyzeService;
    private final DeathClaimAutoNumberService deathClaimAutoNumberService;
    private final DeathClaimDetailService deathClaimDetailService;
    private final DeathClaimCountService deathClaimCountService;
    private final DeathClaimListService deathClaimListService;

    public DeathClaimService(DeathClaimMapper deathClaimMapper,
                           DeathClaimCountService deathClaimCountService,
                           DeathClaimPlaceCountService deathClaimPlaceCountService,
                           DeathClaimAnalyzeService deathClaimAnalyzeService,
                           DeathClaimAutoNumberService deathClaimAutoNumberService,
                           DeathClaimDetailService deathClaimDetailService) {
        this.deathClaimMapper = deathClaimMapper;
        this.deathClaimCountService = deathClaimCountService;
        this.deathClaimPlaceCountService = deathClaimPlaceCountService;
        this.deathClaimAnalyzeService = deathClaimAnalyzeService;
        this.deathClaimAutoNumberService = deathClaimAutoNumberService;
        this.deathClaimDetailService = deathClaimDetailService;
        this.deathClaimCountService = deathClaimCountService;
        this.deathClaimListService = deathClaimListService;
    }

    public List<DeathClaimDTO> getDeathClaims(DeathClaimSearchAndPageDTO sepDTO) {
        return deathClaimMapper.allSelect(sepDTO);
    }

    public int getDeathClaimCount(String searchWord) {
        return deathClaimCountService.selectCount(searchWord);
    }

    public List<DeathClaimCountDTO> getDeathClaimCounts() {
        return deathClaimPlaceCountService.selectPlaceCount();
    }

    public DeathClaimDTO getDeathClaimDetail(String claimNum) {
        return deathClaimDetailService.getDeathClaimDetail(claimNum);
    }

    public void updateDeathClaim(DeathClaimDTO deathClaim) {
        deathClaimMapper.updateDeathClaim(deathClaim);
    }

    public void insertDeathClaim(DeathClaimDTO deathClaim) {
        deathClaimMapper.insertDeathClaim(deathClaim);
    }

    public void deleteDeathClaim(String claimNum) {
        deathClaimMapper.deleteDeathClaim(claimNum);
    }
}
```

VIEW

```
package fdsprojectteam.view;

import fdsprojectteam.domain.DeathClaimCountDTO;
import fdsprojectteam.domain.DeathClaimDTO;
import fdsprojectteam.domain.DeathClaimPlaceCountDTO;
import fdsprojectteam.domain.DeathClaimSearchAndPageDTO;
import org.apache.ibatis.annotations.Param;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.view.RedirectView;

import java.util.List;
import java.util.Map;

@Controller
public class DeathClaimController {
    private final DeathClaimListService deathClaimListService;
    private final DeathClaimDetailService deathClaimDetailService;
    private final DeathClaimCountService deathClaimCountService;
    private final DeathClaimPlaceCountService deathClaimPlaceCountService;
    private final DeathClaimAnalyzeService deathClaimAnalyzeService;
    private final DeathClaimAutoNumberService deathClaimAutoNumberService;
    private final DeathClaimDetailService deathClaimDetailService;
    private final DeathClaimCountService deathClaimCountService;
    private final DeathClaimListService deathClaimListService;

    public DeathClaimController(DeathClaimListService deathClaimListService,
                               DeathClaimDetailService deathClaimDetailService,
                               DeathClaimCountService deathClaimCountService,
                               DeathClaimPlaceCountService deathClaimPlaceCountService,
                               DeathClaimAnalyzeService deathClaimAnalyzeService,
                               DeathClaimAutoNumberService deathClaimAutoNumberService,
                               DeathClaimDetailService deathClaimDetailService,
                               DeathClaimCountService deathClaimCountService,
                               DeathClaimListService deathClaimListService) {
        this.deathClaimListService = deathClaimListService;
        this.deathClaimDetailService = deathClaimDetailService;
        this.deathClaimCountService = deathClaimCountService;
        this.deathClaimPlaceCountService = deathClaimPlaceCountService;
        this.deathClaimAnalyzeService = deathClaimAnalyzeService;
        this.deathClaimAutoNumberService = deathClaimAutoNumberService;
        this.deathClaimDetailService = deathClaimDetailService;
        this.deathClaimCountService = deathClaimCountService;
        this.deathClaimListService = deathClaimListService;
    }

    @GetMapping("deathClaimForm")
    public String deathClaimForm(Model model) {
        return "thymeleaf/deathClaim/deathClaimForm";
    }

    @PostMapping("deathClaimWrite")
    public String deathClaimWrite(@Validated DeathClaimCommand deathClaimCommand, BindingResult result, Model model) {
        if(result.hasErrors()) {
            deathClaimAnalyzeService.execute(model);
        }
        deathClaimListService.insertDeathClaim(deathClaimCommand);
        return "redirect:/deathClaimList";
    }

    @GetMapping("deathClaimList")
    public ModelAndView deathClaimList(@RequestParam("searchWord") String searchWord, Model model) {
        List<DeathClaimCountDTO> counts = deathClaimPlaceCountService.selectPlaceCount();
        Map<String, Object> map = new HashMap<>();
        map.put("counts", counts);
        map.put("searchWord", searchWord);
        return new ModelAndView("thymeleaf/deathClaim/deathClaimList", map);
    }

    @GetMapping("deathClaimDetail")
    public String deathClaimDetail(@Param("claimNum") String claimNum, Model model) {
        DeathClaimDTO detail = deathClaimDetailService.getDeathClaimDetail(claimNum);
        model.addAttribute("detail", detail);
        return "thymeleaf/deathClaim/deathClaimDetail";
    }

    @GetMapping("deathClaimWrite")
    public String deathClaimWrite(Model model) {
        return "thymeleaf/deathClaim/deathClaimWrite";
    }

    @GetMapping("deathClaimList")
    public String deathClaimList(Model model) {
        return "thymeleaf/deathClaim/deathClaimList";
    }
}
```

REPOSITORY

```
package fdsprojectteam.mapper;

import fdsprojectteam.domain.DeathClaimCountDTO;
import fdsprojectteam.domain.DeathClaimDTO;
import fdsprojectteam.domain.DeathClaimPlaceCountDTO;
import fdsprojectteam.domain.DeathClaimSearchAndPageDTO;
import org.apache.ibatis.annotations.Param;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Map;

@Repository
public interface DeathClaimMapper {
    List<DeathClaimDTO> allSelect(DeathClaimSearchAndPageDTO sepDTO);
    int deathClaimCount(String searchWord);
    List<DeathClaimCountDTO> selectCount();
    List<DeathClaimPlaceCountDTO> selectPlaceCount();
    DeathClaimDTO selectionOne(String claimNum);
}
```

CONTROLLER

```
package fdsprojectteam.controller;

import fdsprojectteam.domain.DeathClaimCountDTO;
import fdsprojectteam.domain.DeathClaimDTO;
import fdsprojectteam.domain.DeathClaimPlaceCountDTO;
import fdsprojectteam.domain.DeathClaimSearchAndPageDTO;
import org.apache.ibatis.annotations.Param;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.view.RedirectView;

import java.util.List;
import java.util.Map;

@Controller
public class DeathClaimController {
    private final DeathClaimListService deathClaimListService;
    private final DeathClaimDetailService deathClaimDetailService;
    private final DeathClaimCountService deathClaimCountService;
    private final DeathClaimPlaceCountService deathClaimPlaceCountService;
    private final DeathClaimAnalyzeService deathClaimAnalyzeService;
    private final DeathClaimAutoNumberService deathClaimAutoNumberService;
    private final DeathClaimDetailService deathClaimDetailService;
    private final DeathClaimCountService deathClaimCountService;
    private final DeathClaimListService deathClaimListService;

    public DeathClaimController(DeathClaimListService deathClaimListService,
                               DeathClaimDetailService deathClaimDetailService,
                               DeathClaimCountService deathClaimCountService,
                               DeathClaimPlaceCountService deathClaimPlaceCountService,
                               DeathClaimAnalyzeService deathClaimAnalyzeService,
                               DeathClaimAutoNumberService deathClaimAutoNumberService,
                               DeathClaimDetailService deathClaimDetailService,
                               DeathClaimCountService deathClaimCountService,
                               DeathClaimListService deathClaimListService) {
        this.deathClaimListService = deathClaimListService;
        this.deathClaimDetailService = deathClaimDetailService;
        this.deathClaimCountService = deathClaimCountService;
        this.deathClaimPlaceCountService = deathClaimPlaceCountService;
        this.deathClaimAnalyzeService = deathClaimAnalyzeService;
        this.deathClaimAutoNumberService = deathClaimAutoNumberService;
        this.deathClaimDetailService = deathClaimDetailService;
        this.deathClaimCountService = deathClaimCountService;
        this.deathClaimListService = deathClaimListService;
    }

    @GetMapping("deathClaimForm")
    public String deathClaimForm(Model model) {
        return "thymeleaf/deathClaim/deathClaimForm";
    }

    @PostMapping("deathClaimWrite")
    public String deathClaimWrite(@Validated DeathClaimCommand deathClaimCommand, BindingResult result, Model model) {
        if(result.hasErrors()) {
            deathClaimAnalyzeService.execute(model);
        }
        deathClaimListService.insertDeathClaim(deathClaimCommand);
        return "redirect:/deathClaimList";
    }

    @GetMapping("deathClaimList")
    public ModelAndView deathClaimList(@RequestParam("searchWord") String searchWord, Model model) {
        List<DeathClaimCountDTO> counts = deathClaimPlaceCountService.selectPlaceCount();
        Map<String, Object> map = new HashMap<>();
        map.put("counts", counts);
        map.put("searchWord", searchWord);
        return new ModelAndView("thymeleaf/deathClaim/deathClaimList", map);
    }

    @GetMapping("deathClaimDetail")
    public String deathClaimDetail(@Param("claimNum") String claimNum, Model model) {
        DeathClaimDTO detail = deathClaimDetailService.getDeathClaimDetail(claimNum);
        model.addAttribute("detail", detail);
        return "thymeleaf/deathClaim/deathClaimDetail";
    }

    @GetMapping("deathClaimWrite")
    public String deathClaimWrite(Model model) {
        return "thymeleaf/deathClaim/deathClaimWrite";
    }

    @GetMapping("deathClaimList")
    public String deathClaimList(Model model) {
        return "thymeleaf/deathClaim/deathClaimList";
    }
}
```

14. 소스코드

자동차보험

```
<!DOCTYPE html> kimyonghyuk, 6/11/24, 9:20 오전 • [FEAT] 자동차 보험 시나리오 완성본
<html>
<head>
<meta charset="UTF-8">
<title>carClaimForm.html</title>
<link rel="stylesheet" type="text/css" href="../static/bootstrap/bootstrap.min.css">
</head>
<body>
<br />
<h3 style="...">자동차보험 청구서</h3>
<br />
<form action="carClaimWrite" method="post">





```

VIEW

SERVICE

CONTROLLER

```
@Service 3 usages kimyonghyuk
public class CarClaimWriteService { kimyonghyuk, 6/11/24, 9:20 오전 • [FEAT] 자동차 보험 시나리오 완성본
    @Autowired
    CarClaimMapper carClaimMapper;
    public List<String> execute(CarClaimCommand carClaimCommand, Model model) { kimyonghyuk

        // 차량 정보 목록 번호 또는 모델명과 최대 탑승 인원을 예상
        final Map<String, Integer> VEHICLE_CAPACITY = Map.of(
            "k1: 현대 아반떼", v1: 5,
            "k2: 기아 K5", v2: 5,
            "k3: 쉐보레 캠aro", v3: 5,
            "k4: 현대 틀리세이드", v4: 7,
            "k5: 혼다 피트", v5: 7,
            "k6: 포드 익스플로러", v6: 7,
            "k7: 기아 쏘렌토", v7: 7,
            "k8: 현대 스타리아", v8: 9,
            "k9: 기아 카니발", v9: 9,
            "k10: 포드 트랜싯", v10: 9
        );

        // 7명 이상 탑승 수 있는 차량 목록
        final List<String> LARGE_CAPACITY_VEHICLES = List.of("현대 틀리세이드", "혼다 피트", "포드 익스플로러", "기아 쏘렌토", "현대 틀리세이드");
        // 평균 대형 차량 목록 (당시 로직에 사용함)
        final List<String> LARGE_VEHICLE_TYPES = List.of("현대 스타리아", "기아 카니발", "포드 트랜싯");

        // 데이터베이스 DTO로 변환
        CarClaimDTO dto = new CarClaimDTO();
        dto.setClaimAccidentDate(carClaimCommand.getClaimAccidentDate());
        dto.setClaimAddr(carClaimCommand.getClaimAddr());
        dto.setClaimCar(carClaimCommand.getClaimCar());
        dto.setClaimCount(carClaimCommand.getClaimCount());
        dto.setClaimHospital(carClaimCommand.getClaimHospital());
        dto.setClaimLicenseNumber(carClaimCommand.getClaimLicenseNumber());
        dto.setClaimLicenseType(carClaimCommand.getClaimLicenseType());
        dto.setClaimLocation(carClaimCommand.getClaimLocation());
        dto.setClaimName(carClaimCommand.getClaimName());
        dto.setClaimNumber(carClaimCommand.getClaimNumber());
        dto.setClaimSignName(carClaimCommand.getClaimSignName());
        dto.setClaimSituation(carClaimCommand.getClaimSituation());
        dto.setClaimTel(carClaimCommand.getClaimTel());
        dto.setMemCarNumber(carClaimCommand.getMemCarNumber());
        model.addAttribute("carClaimCommand", carClaimCommand);
        // carClaimMapper.carClaimInsert(dto);
        List<String> resultMessages = new ArrayList<>();

        // 다수인 탑승자인원수 팀지 (고의사고 가능성)
        if (dto.getClaimCount() > 7) {
            model.addAttribute("attributeName: "success", attributeValue: false);
            resultMessages.add("탑승자 인원수가 많아서 클레임이 보류되었습니다.");
        }

        if (LARGE_CAPACITY_VEHICLES.contains(dto.getClaimCar()) || dto.getClaimCount() > 7) {
            model.addAttribute("attributeName: "success", attributeValue: false);
            resultMessages.add("해당 차량은 7명 이상 탑승할 수 있으므로 클레임이 보류되었습니다.");
        }

        Integer maxCapacity = VEHICLE_CAPACITY.get(dto.getClaimCar());
        if (maxCapacity != null && dto.getClaimCount() > maxCapacity) {
            model.addAttribute("attributeName: "success", attributeValue: false);
            resultMessages.add("차량의 정원(" + maxCapacity + "명) 이상 인원이 탑승하여 클레임이 보류되었습니다.");
        }

        if (LARGE_VEHICLE_TYPES.contains(dto.getClaimCar())) {
            model.addAttribute("attributeName: "success", attributeValue: false);
            resultMessages.add("해당 차량 차종(" + dto.getClaimCar() + ")은 대형 차량으로 클레임이 보류되었습니다.");
        }

        //
        // 5. 일전하고 비슷한 사고경위일 경우 보류
        List<CarClaimDTO> similarClaims = carClaimMapper.findSimilarClaims(dto.getClaimSituation());
        if (!similarClaims.isEmpty()) {
            model.addAttribute("attributeName: "success", attributeValue: false);
            resultMessages.add("이전 사고와 비슷한 사고경위로 인해 클레임이 보류되었습니다.");
        }

        List<CarClaimDTO> recentClaims = carClaimMapper.findRecentClaims(params);
        if (!recentClaims.isEmpty()) {
            model.addAttribute("attributeName: "success", attributeValue: false);
            resultMessages.add("최근 6개월 내에 사고 제출 여부 확인");
        }

        // 탑승자로의 걸리지 않을경우
        if (resultMessages.isEmpty()) {
            carClaimMapper.carClaimInsert(dto);
            model.addAttribute("attributeName: "success", attributeValue: true);
            resultMessages.add("클레임이 성공적으로 처리되었습니다.");
        }

        return resultMessages;
    }
}

@Mapper 3 usages kimyonghyuk
public interface CarClaimMapper {
    public void carClaimInsert(CarClaimDTO dto); 1 usage kimyonghyuk
    List<CarClaimDTO> findSimilarClaims(@Param("claimSituation") String claimSituation); 1 usage kimyonghyuk
    List<CarClaimDTO> findRecentClaims(Map<String, Object> params); 1 usage kimyonghyuk
}

<?xml version="1.0" encoding="UTF-8"?> kimyonghyuk, 6/11/24, 9:20 오전 • [FEAT] 자동차 보험 시나리오 완성본
<!DOCTYPE mapper PUBLIC
    "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="fdsprojectteam.mapper.CarClaimMapper">
<insert id="carClaimInsert" parameterType="carClaim">
    insert into carClaim(claim_number,mem_carnumber,claim_name,claim_car,claim_tel
    ,claim_licensenum,claim_licensetype,claim_accidentdate,claim_location,claim_situation
    ,claim_count,claim_hospital,claim_signname,claim_addr)
    values(#{claimNumber},#{memCarNumber},#{claimName}
    ,#{claimCar},#{claimTel},#{claimlicenseNumber},#{claimLicenseType}
    ,#{claimAccidentDate},#{claimLocation},#{claimSituation},#{claimCount},#{claimHospital},#{claimSignName},#{claimAddr})</insert>
    <!-- 유사한 사고 조회 -->
    <select id="findSimilarClaims" resultType="fdsprojectteam.domain.CarClaimDTO">
        SELECT *
        FROM carclaim
        WHERE claim_situation = #{claimSituation}</select>
    <select id="findRecentClaims" parameterType="map" resultType="fdsprojectteam.domain.CarClaimDTO">
        SELECT *
        FROM CarClaim
        WHERE mem_carnumber = #{memCarNumber}
        AND claim_accidentdate >= TRUNC(ADD_MONTHS(#{claimAccidentDate}, -6), 'MM') -- 6개월 전의 월의 첫 날짜
        AND claim_accidentdate < TRUNC(#{claimAccidentDate}, 'MM') + 1 -- 입력된 월의 첫 날짜
    </select>
</mapper>
```

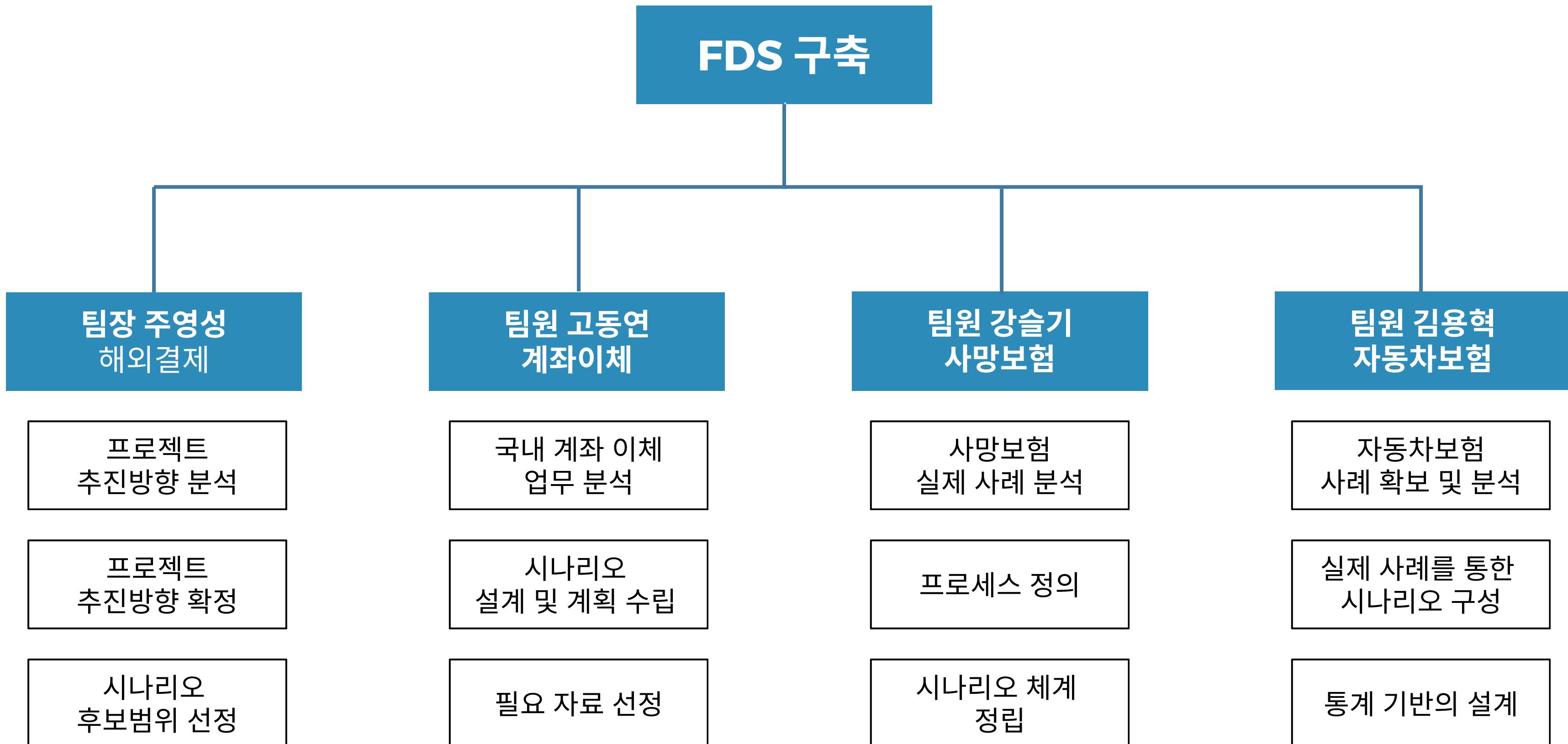
REPOSITORY

MAPPER

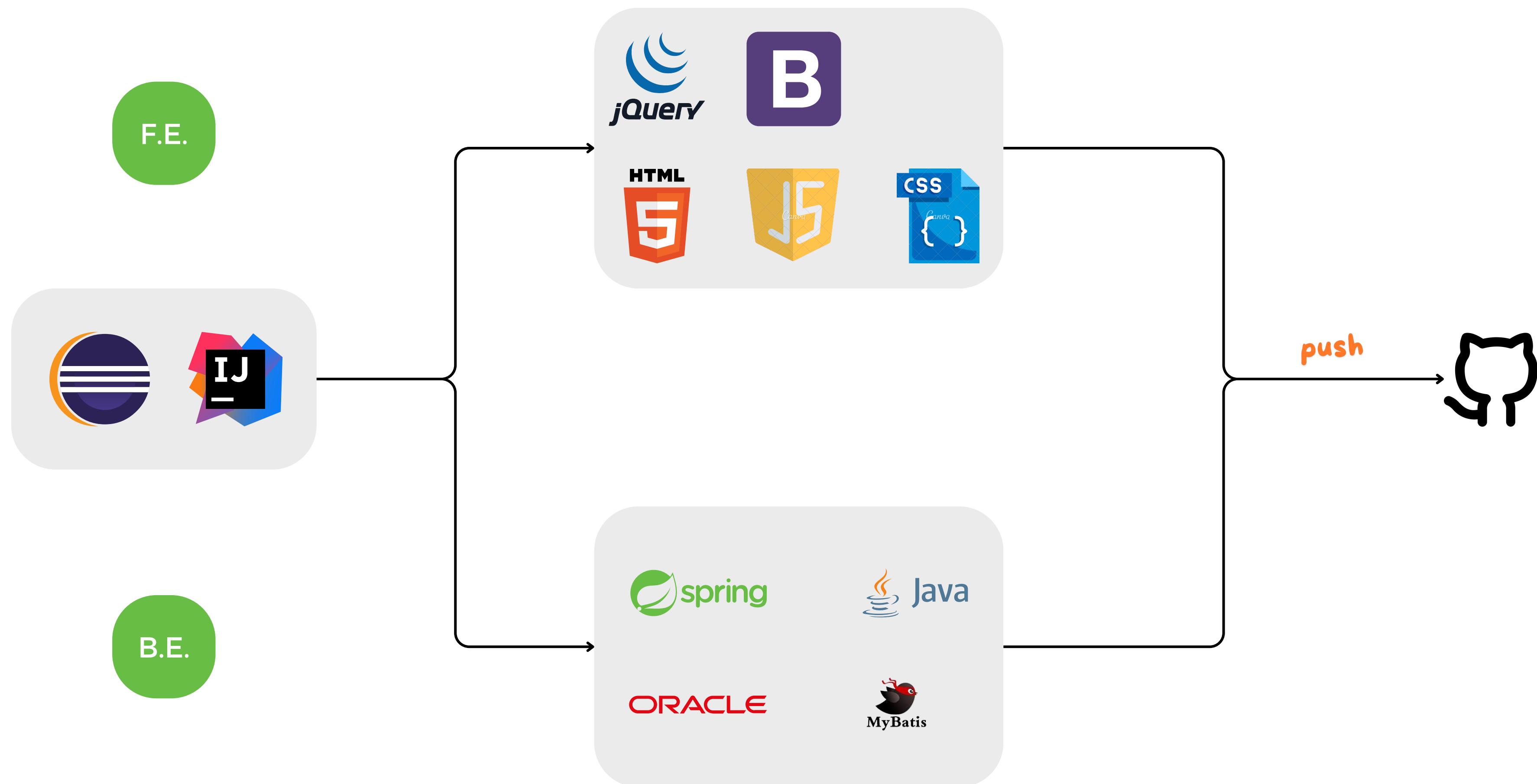
15. 일정(WBS)

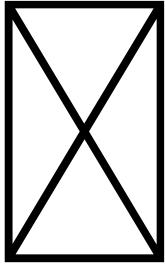
단위	진행목록	진행자	5월 14일 ~ 6월 26일					
			1주차	2주차	3주차	4주차	5주차	6주차
개념설계	FDS관련 적용 시제 검색	팀전체						
	FDS 아이템 선정							
	시나리오 구성							
	시나리오 명사들을							
	면티피 태입 정의							
	ERD 관계 정의							
	속성 정의							
	식별자 정의							
	도메인 정의							
논리설계	정규화							
	데이터 정의							
프로젝트 준비	GitHub Session	주행선						
	프로젝트 기본 세팅 구축							
	Index Page 구성							
	PPT 구성	팀전체						
	발표자료							
	DB구성 및 Mock데이터 삽입							
개인 회의	개인 회의	고용연						
	조회 결과							
해외 카드 편집	결제 품목	주행선						
	비정상 거래 고객 리스트							
	고객 페이지							
	비정상 거래 고객 상세 페이지							
사용자 모임	사용보통 예전 페이지	접속자						
	청구서 품목							
	청구서 조회							
	지급 결과 확인							
자동화 모듈	청구서 합체	접속자						
	청구서 출판							
	자동화 모듈 차트							

16. 조직도(WBS)



17. 사용된 프로그램





The End

감사합니다