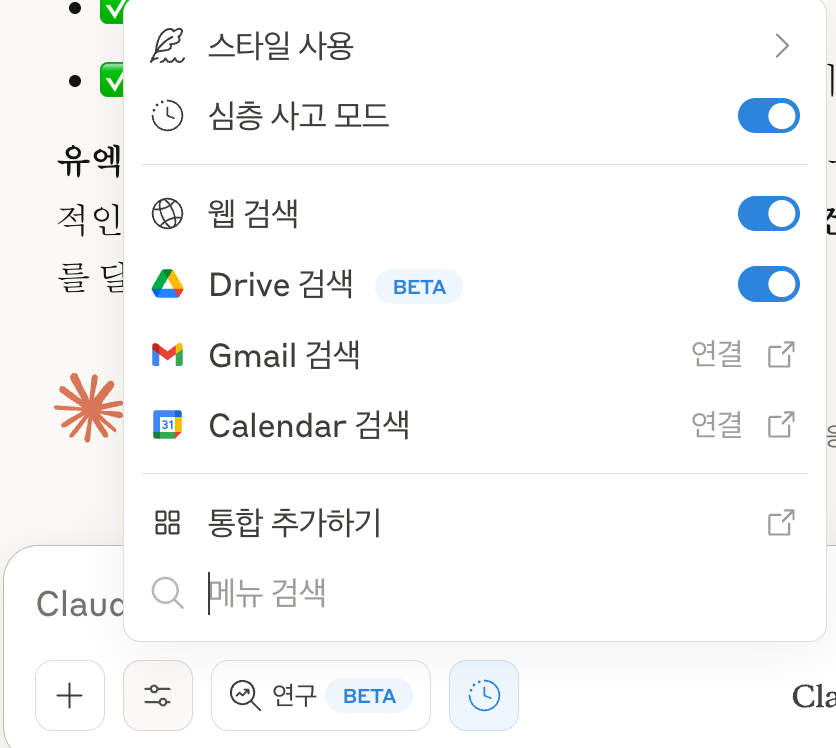
서비스 기획 가이드

# 1. 문제정의와 솔루션 선정

<https://github.com/cna-bootcamp/aiguide/blob/main/AI%ED%99%9C%EC%9A%A9%20%EC%84%9C%EB%B9%84%EC%8A%A4%20%EA%B8%B0%ED%9A%8D%20%EA%B0%80%EC%9D%B4%EB%93%9C.md#%EC%83%9D%EC%84%B1%ED%98%95ai%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%9C-%EC%84%9C%EB%B9%84%EC%8A%A4-%EA%B8%B0%ED%9A%8D-%EC%8B%A4%EC%8A%B5-%EA%B0%80%EC%9D%B4%EB%93%9C>

# 2. 기획 구체화: Event Storming

프롬프트 옵션에서 ‘심층 사고 모드’를 활성화 하여 수행함.



## 1) **Event 도출 프롬프트**

Knowledge에 ‘서비스기획서’라는 이름으로 gamma.app에 요청할 때 사용한 텍스트를 저장함.

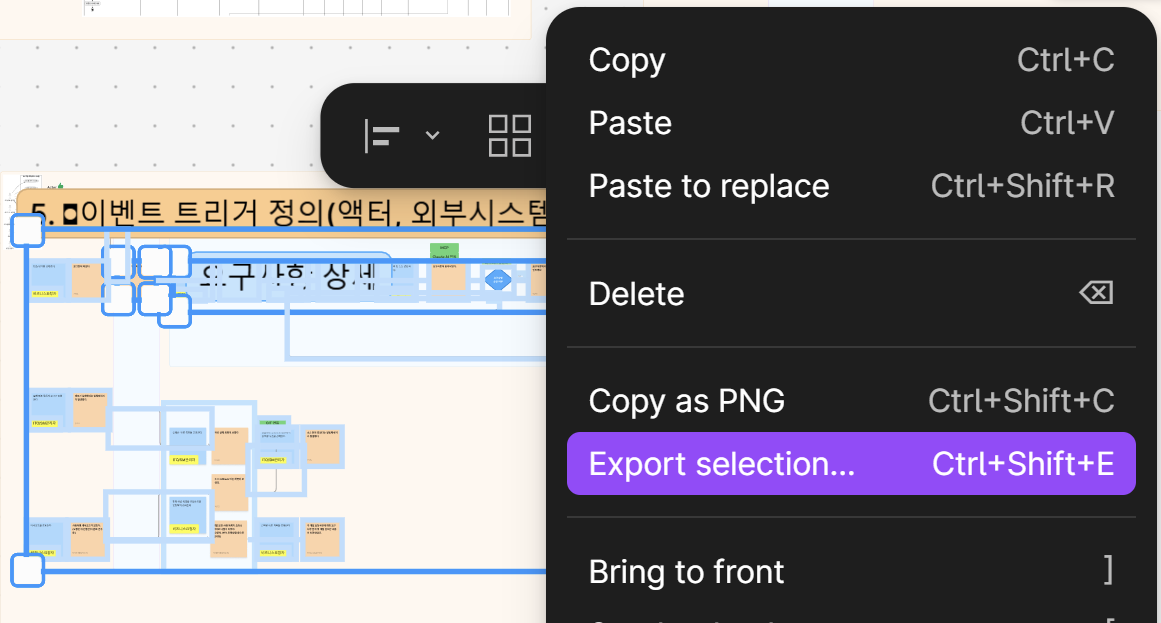
또는 gamma.app의 결과를 ‘서비스기획서.pdf’로 내보낸 후 Knowledge에 업로드함.

| o: 마이크로서비스로 설계하기 위해 DDD전략설계를 하고자 합니다.  이를 위해 Event Storming을 이용하고자 합니다.  [요청사항]  - 각 자 Event를 도출하고 정리  - 표준 plantuml script로 Activity 다이어그램 제작  [참고자료]  서비스기획서  [응답형식]  - 코드블록 |
| --- |

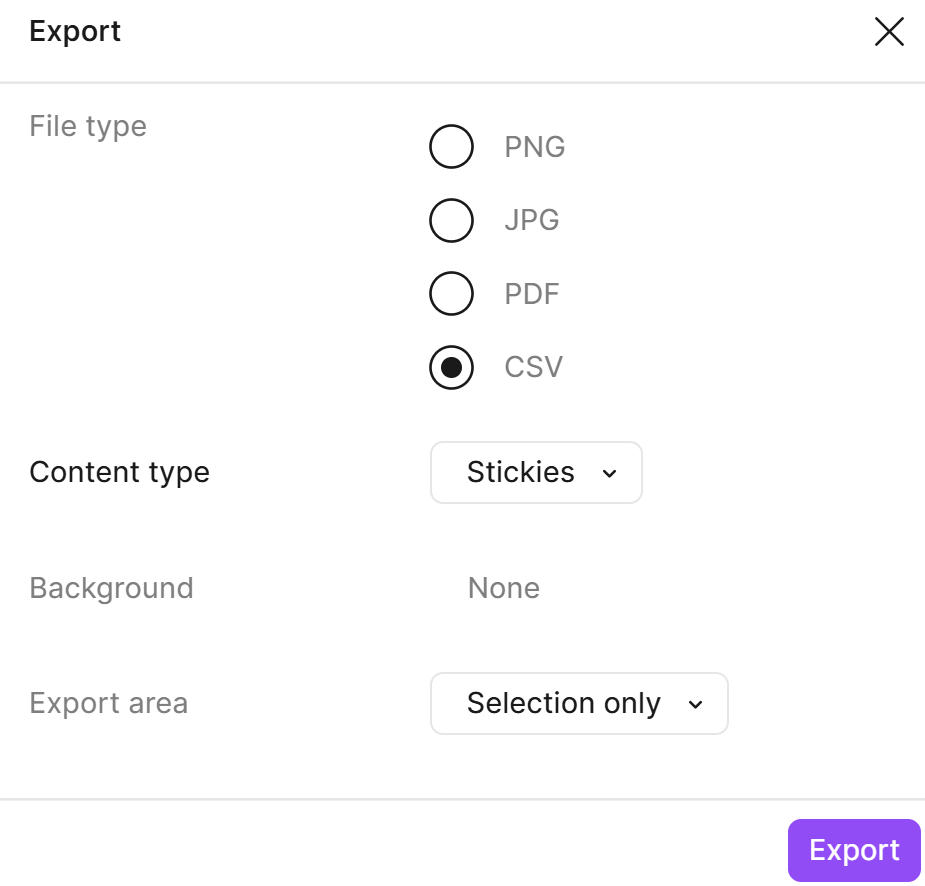
## 2) Command 정의

- Figma에서 아래와 같이 Event를 Export함.

스티커 선택 후 우측 마우스 메뉴에서 ‘Export selection’ 선택



CSV 형식으로 Export



다운로드한 CSV파일을 NotePad++에서 오픈한 후 내용을 복사함.

Knowledge에 ‘이벤트’라는 제목으로 업로드.

Command 정의를 요청함

| o: Command 정의  [요청사항]  - 각 이벤트를 발생시키는 Command를 정의  - Command와 Event를 모두 표시  - 표준 plantuml script로 Activity 다이어그램 제작  - 스타일 지정 사용  ```  예: skinparam activityBackgroundColor<<command>> #FFB570와 같은 방식으로  그리고 활동에 스테레오타입 적용: :회원가입 요청 <<command>>;  ```  [참고자료]  - 서비스기획서  - 이벤트  [응답형식]  - 코드블록 |
| --- |

## 3) Actor 정의

동일한 대화창에서 요청함

| o: Actor  정의  [요청사항]  - Command를 수행하는 Actor 정의  - 각 Actor가 수행하는 Command를 표시  - 표준 PlantUML Script로 Usecase 다이어그램 작성  [참고자료]  - 서비스기획서  - 도출된 Command  [응답형식]  - 코드블록 |
| --- |

## 4) Policy/Rule 정의

동일한 대화창에서 요청함.

핵심 이벤트에 대해서 Policy와 Rule 정의 요청

예시)

| 아래 이벤트의 Policy와 Rule을 정의해 주세요.  - ‘매장 정보’ 항목으로 사진과 메뉴 내용이 표시되었고, 편집 메뉴가 표시되었다  - 마케팅 현황이 보였다 |
| --- |

## 5) Entity 정의

지금까지 정의한 Event, Command, Actor, Policy/Rule, External System을 Figma에서 선택 후

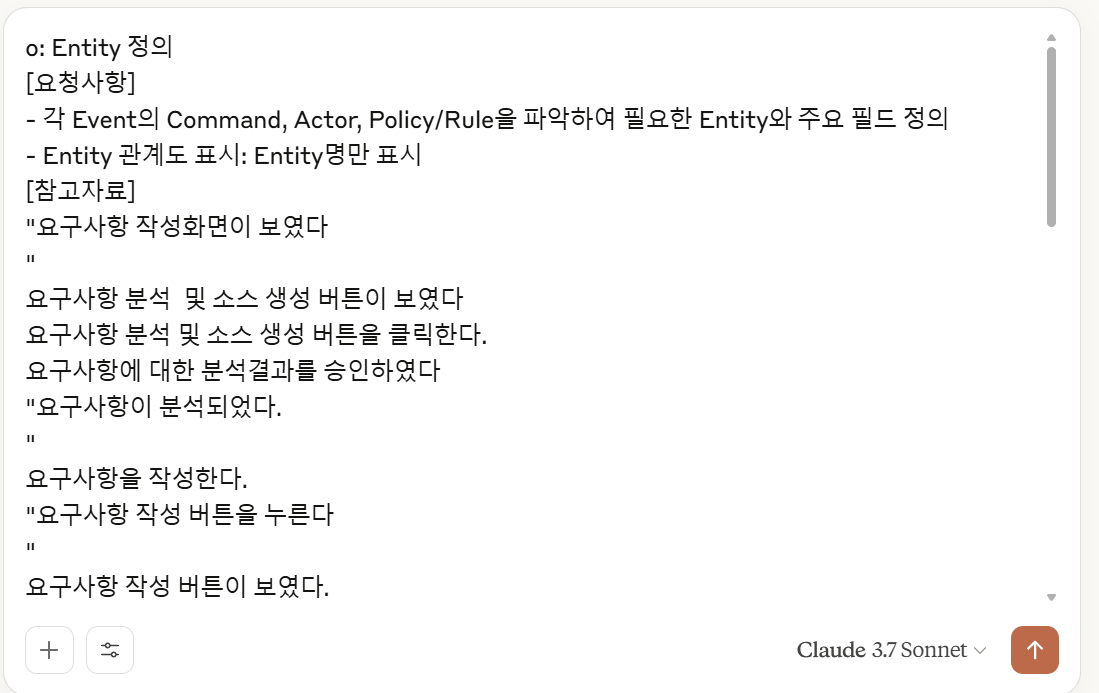
Export합니다.

다운로드 한 CSV파일을 Notepad++에서 엽니다. 내용을 복사합니다.

그리고 ‘참고자료’로 제시 합니다.

| o: Entity 정의  [요청사항]  - 각 Event의 Command, Actor, Policy/Rule을 파악하여 필요한 Entity와 주요 필드 정의  - Entity 관계도 표시: Entity명만 표시  - Entity관계는 plantuml script로 Entity Relationship Diagrams으로 작성  - 각 Entity명과 관계를 한글로 표시  - 1:1, 1:N, N:N 등의 관계 표시  [참고자료]  {Figma에서 Export한 데이터} |
| --- |

예시)



## 6) Aggregate정의

Entity도출과 동일한 대화창에서 요청함

| o: Aggregate 정의  [요청사항]  - Entity 관계도를 분석하여 Aggregate 정의  - Aggregate간의 관계는 plantuml script로 Diagrams 작성  - Aggregate명과 관계를 한글로 표시  - Entity의 프라퍼티와 메소드까지 표시 |
| --- |

## 7) Bounded Context 정의

Aggregate정의와 동일한 대화창에서 요청함

| o: Bounded Context 정의  [요청사항]  - Aggregate정의 결과를 분석하여 Bounded Context 정의  - Bounded Context간의 관계를 plantuml script로 Diagrams 작성  - Bounded Context명과 관계를 한글로 표시  - note는 표시하지 말것 |
| --- |

## 8) Context Map 작성

계속 동일한 대화창에서 요청함.

| o: Context Map 작성  [요청사항]  - Bounded Context와 External System만 이용하여 작성  - Bounded Context의 Entity는 표시하지 않음  - DDD관계패턴은 영어로 표시하고 ‘<<DDD관계 패턴>>’형식으로 표시  - 각 Bounded Context와 External System간의 관계는 한글로 사용자 친화적으로 쉽게 표시  - 1:1, 1:N, N:N 등의 관계가 아닌 화살표로 참조 방향 표시  - note는 생략하여 다이어그램 단순화 |
| --- |

External System과의 관계가 Loosely Coupling되었는지 검토 요청함

| o: External System과의 관계가 Loosely coupling 되었는지 검토해 주세요.  [참고자료]  - Cloud Design Patterns(개요).docx |
| --- |

개선된 관계를 반영하여 다시 Context Map을 작성하도록 요청함

| o: 개선안을 반영하여 Context Map 재작성  [요청사항]  - 타이틀은 ‘Context Map’으로 함  - Bounded Context와 External System, Message Queue 인프라를 모두 포함하여 작성  - Bounded Context명은 한글로 표시하고 동일한 색깔로 구별  - Message Queue는 주황색(#orange)으로 표시  - Cloud Design Patterns 보호 계층 포함 (예: Anti-Corruption Layer, Circuit Breaker 등)  - DDD관계패턴은 영어로 표시하고 '<<DDD관계 패턴>>'형식으로 표시  - 각 관계는 한글로 사용자 친화적이고 비즈니스 가치 중심으로 표시  - 비동기 통신 패턴 표시(예: Publisher-Subscriber, CQRS 등)  - Message Queue를 통한 이벤트 기반 통신 흐름 표시  - External System과의 Loosely Coupled 관계 표시 (예: Conformist, Anti-Corruption Layer 등)  - 화살표로 참조 방향 표시, note는 생략하여 다이어그램 단순화  - PlantUML !theme mono 사용 |
| --- |

DDD관계패턴을 생략하여 더 간소하게 제작

| DDD관계패턴을 생략하여 Context Map을 다시 그려줘요. |
| --- |