스마트 출입 솔루션

소프트웨어 아키텍처 명세

작성자: 이유훈

Revision History

|  |  |  |
| --- | --- | --- |
| Version | Date | Summary |
| 1 | 9/14 | Business Goals 작성 |
| 2 | 9/17 | Business Constraints 작성 |
| 3 | 10/01 | 시장 조사 및 BG 재설정 |
| 4 | 10/04 | BG, BC 재작성 |
|  |  |  |
|  |  |  |
|  |  |  |

내용

[1. Business Drivers 5](#_Toc200455366)

[1.1. Business Goals 5](#_Toc200455367)

[1.1.1. Stakeholders List 5](#_Toc200455368)

[1.1.2. Business Goals 5](#_Toc200455369)

[1.2. Business Constraints 6](#_Toc200455370)

[2. System Context 7](#_Toc200455371)

[2.1. System Context Diagram 7](#_Toc200455372)

[2.2. External Entity List 7](#_Toc200455373)

[2.3. External Interface List 8](#_Toc200455374)

[3. Architectural Drivers 10](#_Toc200455375)

[3.1. Use Case Model 10](#_Toc200455376)

[3.1.1. Use Case Diagram 10](#_Toc200455377)

[3.1.2. Actor List 11](#_Toc200455378)

[3.1.3. Use Case List 11](#_Toc200455379)

[3.1.4. UC-01 Title Description 12](#_Toc200455380)

[3.1.5. UC-02 Title Description 13](#_Toc200455381)

[3.1.6. *UC-03 Title* Description 13](#_Toc200455382)

[3.2. Quality Attribute Scenario 13](#_Toc200455383)

[3.2.1. QA Scenario List 13](#_Toc200455384)

[3.2.2. QA-01 Title Scenario 13](#_Toc200455385)

[3.2.3. QA-02 Title Scenario 14](#_Toc200455386)

[3.3. Architectural Constraint 14](#_Toc200455387)

[4. High Level Structure Design Description 16](#_Toc200455388)

[4.1. Domain Modeling 16](#_Toc200455389)

[4.1.1. Conceptual Class List 16](#_Toc200455390)

[4.1.2. Dynamic View 16](#_Toc200455391)

[4.1.3. Static View 17](#_Toc200455392)

[4.2. Quality Driven Architectural Design 18](#_Toc200455393)

[4.2.1. Exploring Architectural Options for *QA-01 Title* 18](#_Toc200455394)

[4.2.2. Exploring Architectural Options for *QA-02 Title* 20](#_Toc200455395)

[4.3. Component & Connector View 20](#_Toc200455396)

[4.3.1. UML Component Diagram 21](#_Toc200455397)

[4.3.2. Component List 21](#_Toc200455398)

[4.3.3. Connector List 22](#_Toc200455399)

[4.4. Deployment View 23](#_Toc200455400)

[4.4.1. Deployment Diagram 23](#_Toc200455401)

[4.4.2. Artifact Definition Diagram 25](#_Toc200455402)

[5. Detailed Component Design Description 27](#_Toc200455403)

[5.1. Design Description for *ComponentName* 27](#_Toc200455404)

[5.1.1. Overview 27](#_Toc200455405)

[5.1.2. Quality Driven Component Design 28](#_Toc200455406)

[5.1.3. Module View 30](#_Toc200455407)

[5.2. Design Description for *Component 2* 32](#_Toc200455408)

[5.3. Design Description for *Component 3* 32](#_Toc200455409)

[6. Architectural Evaluation 33](#_Toc200455410)

[*6.1.* Architectural Evaluation for *QA-01 Title* 33](#_Toc200455411)

[6.1.1. List of Risks 33](#_Toc200455412)

[6.1.2. List of Sensitivities 33](#_Toc200455413)

[6.1.3. List of Tradeoffs 34](#_Toc200455414)

[6.1.4. List of Nonrisks 34](#_Toc200455415)

[7. Appendix 35](#_Toc200455416)

[7.1. Detailed Component Specification for C&C View 35](#_Toc200455417)

[7.1.1. Component 1 Name 35](#_Toc200455418)

[7.1.2. Component 2 Name 36](#_Toc200455419)

[7.2. Interface specifications for C&C View 36](#_Toc200455420)

[7.2.1. *Interface2\_1* Interface Specification 36](#_Toc200455421)

# Business Drivers

본 장에서는 스마트 출입 솔루션의 이해관계자, 비즈니스 목표, 비즈니스 제약사항에 대해 설명합니다. 본 시스템은 에어비앤비 등 단기 임대 플랫폼을 사용하는 개인 호스트를 주 타겟으로, 예약 정보를 자동으로 동기화하여 게스트를 위한 기간 한정 임시 출입 코드를 생성 및 관리해주는 클라우드 기반 자동화 플랫폼입니다. 이를 통해 호스트는 반복적인 출입 관리 업무에서 해방되고, 게스트는 마찰 없는 체크인 경험을 누리게 됩니다.

## Business Goals

### Stakeholders List

|  |  |
| --- | --- |
| 이해관계자 | 역할 및 관심사항 |
| 게스트 (Guest) | [역할] 숙소에 정해진 시간 동안 머무는 최종 사용자 [관심사항] 체크인 시간에 맞춰 쉽고 빠르게 숙소에 출입하는 것, 복잡한 절차 없는 직관적인 출입 방식, 호스트와의 불필요한 소통 최소화 |
| 에어비앤비 개인 호스트 | [역할] 1개 ~ 소수(2-10개)의 숙소를 운영하며, 출입 관리 자동화를 통해 운영 효율성을 높이고자 하는 핵심 고객 [관심사항] 게스트마다 비밀번호를 수동으로 변경하고 전달하는 시간 낭비 해결, 이전 게스트의 무단출입 등 보안 우려 해소, 게스트의 체크인 경험 향상을 통한 긍정적 후기 확보, 저렴하고 사용하기 쉬운 솔루션 도입 |
| 경영진 | [역할] 비즈니스 전략 및 성장 로드맵 수립, 수익 모델 검증, 시장 경쟁력 확보 [관심사항] 틈새시장(개인 호스트)의 빠른 선점, '첫 번째 잠금장치 무료' 모델을 통한 사용자 기반(Critical Mass) 확보, 장기적인 관점에서 공식 플랫폼 파트너십으로 전환할 수 있는 협상력 확보, 안정적인 구독 기반 수익 창출 |
| 개발팀 | [역할] 시스템의 설계, 개발, 테스트 및 배포 [관심사항] iCalendar 표준의 안정적인 파싱 및 동기화, 다양한 스마트락 통합 API(Aggregator)의 안정적 연동, 확장성과 비용 효율성을 고려한 클라우드 아키텍처 구축, 안정적인 서비스 운영 |
| 운영팀 | [역할] 배포된 시스템의 모니터링, 호스트의 기술 지원 및 고객 문의 응대 [관심사항] 시스템 장애 최소화 및 신속한 복구, 호스트가 iCal URL 연동이나 스마트락 계정 연결 시 겪는 문제에 대한 효율적인 지원 체계 마련, 서비스 안정성 유지 |

### Business Goals

|  |  |  |  |
| --- | --- | --- | --- |
| 이해관계자 | 비즈니스 목표 | | |
| ID | 기술 | 중요도\* |
| 경영진, 개인 호스트 | BG-01 | [목표] 출시 2년 내 개인 호스트 자동 출입 관리 시장 점유율 15% 달성 파편화된 개인 호스트 시장에 빠르게 침투하여 초기 시장의 지배적 사업자로 자리매김하고, 이를 바탕으로 주요 예약 플랫폼과의 공식 파트너십 협상력을 확보한다.  [아키텍처와의 연관성] 초기 사용자 확보 전략(Freemium)을 지원하기 위해 다수의 사용자를 비용 효율적으로 수용할 수 있는 아키텍처가 필수적이다. 또한, 사용자 증가에 유연하게 대응할 수 있는 확장성(Scalability)이 요구된다.  [중요도 산정 근거] 시장 점유율은 후발 주자의 진입을 막는 가장 강력한 해자(Moat)이며, 플랫폼 비즈니스의 장기적인 가치와 생존을 결정하는 핵심 지표이므로 가장 높은 중요도를 부여한다. | 5 |
| 개인 호스트, 게스트, 경영진 | BG-02 | [목표] 호스트 만족도 95% 달성을 통한 유료 전환율 10% 확보 호스트의 수동 관리 업무(비밀번호 전달, 변경 등)를 완벽히 제거하여 높은 만족도를 이끌어내고, 이를 통해 무료 사용자를 유료 구독자로 전환시켜 안정적인 초기 매출을 창출한다.  [아키텍처와의 연관성] 외부 시스템(iCal, 스마트락 API)의 일시적 오류에도 불구하고, 예약-출입 코드 생성-만료로 이어지는 핵심 워크플로우가 99.9%의 성공률로 동작하는 높은 신뢰성(Reliability)과 고가용성(High-Availability)을 보장하는 아키텍처가 필요하다.  [중요도 산정 근거] 호스트 만족도는 서비스 이탈률(Churn Rate)을 낮추고, Freemium 모델이 실제 매출로 이어지는지를 결정하는 핵심 선행 지표이다. 신뢰성 없는 자동화는 오히려 만족도를 해치므로 가장 높은 중요도를 부여한다. | 5 |
| 개발팀,  운영팀 | BG-03 | [목표] 자동화된 모니터링 및 장애 복구 시스템 구축을 통한 연간 운영 비용 30% 절감 시스템의 장애 감지, 복구, 리소스 확장 과정을 자동화하여 사람의 수동 개입을 최소화하고, 이를 통해 서비스 안정성을 높이는 동시에 인프라 및 인적 자원 비용을 절감한다.  [아키텍처와의 연관성] 각 컴포넌트의 상태를 실시간으로 감시하는 모니터링 시스템(Heartbeat 등)과, 장애 발생 시 자동으로 서비스를 재시작하거나 트래픽을 우회시키는 자동 복구 메커니즘(Active Redundancy 등)을 아키텍처에 포함하게 된다.  [중요도 산정 근거] 운영 비용 절감은 수익성을 개선하고 사업의 지속 가능성을 높이는 데 직접적으로 기여한다. 또한, 안정적인 서비스 운영은 호스트 만족도(BG-02)의 기본 전제 조건이다. | 4 |
| 경영진, 개발팀 | BG-04 | [목표] 출시 2년 차 프로(Pro) 요금제 도입을 통한 연간 1억 원의 추가 매출 달성 장기 투숙자 및 소규모 숙소 관리자를 대상으로 하는 '앱 기반 출입 권한 부여' 등 프리미엄 기능을 제공하여 새로운 매출원을 확보하고, 인접 시장으로의 확장 가능성을 검증한다.  [아키텍처와의 연관성] 초기 설계부터 다양한 인증 방식(숫자 코드, 앱 권한)과 요금제를 수용할 수 있는 유연한 사용자 및 접근 관리(IAM), 구독 관리 아키텍처를 고려해야 한다.  [중요도 산정 근거] Freemium 모델의 한계를 넘어 지속적으로 성장하기 위해서는, 더 높은 가치를 제공하며 객단가(ARPU)를 높일 수 있는 프리미엄 서비스로의 확장이 필수적이다. | 3 |

## Business Constraints

|  |  |
| --- | --- |
| 비즈니스 제약사항 | |
| ID | 기술 |
| BC-01 | [기술적 제약] 예약 플랫폼 API의 폐쇄성 주요 예약 플랫폼(에어비앤비 등)의 공식 API는 신규 파트너에게 개방되어 있지 않아, 실시간 데이터의 직접 연동이 불가능하다.  [아키텍처 영향] 이 제약은 '완벽한 비대면 체크인 경험 제공(BG-02)'이라는 목표 달성을 위해 MVP 단계에서 iCalendar 동기화 방식의 채택을 강제한다. 아키텍처는 수많은 iCal URL을 주기적으로 폴링하고, 변경 사항을 안정적으로 감지하며, 파싱 오류에 대응할 수 있는 비동기적이고 확장 가능한 데이터 수집 시스템을 반드시 포함해야 한다. |
| BC-02 | [시간 제약] 시장 선점을 위한 빠른 출시 (6개월 내) 경쟁자가 없는 시장 기회를 선점하기 위해 6개월 내에 핵심 기능을 갖춘 최소 기능 제품(MVP)을 출시해야 한다.  [아키텍처 영향] '시장 점유율 확보(BG-01)' 목표를 달성하기 위해, 개발 기간을 단축해야 한다. 따라서 모든 스마트락을 직접 연동하는 대신, Seam.co와 같은 제3자 통합 API(Aggregator)를 활용하고, 인프라 구축 시간을 최소화하기 위해 서버리스(Serverless) 아키텍처와 관리형 클라우드 서비스(Managed Services)를 적극적으로 채택해야 한다. |
| BC-03 | [비용 제약] Freemium 모델 기반의 저비용 운영 초기 사용자 확보 전략인 '첫 번째 잠금장치 무료' 비즈니스 모델을 지원하기 위해, 다수의 무료 사용자를 수용하면서도 총소유비용(TCO)을 최소화해야 한다.  [아키텍처 영향] '운영 비용 절감(BG-03)' 목표와 직결되는 제약으로, 유휴 시간에는 비용이 거의 발생하지 않는 서버리스 아키텍처(예: AWS Lambda)를 설계의 핵심 원칙으로 삼는다. 또한, 트래픽 변화에 따라 자원이 탄력적으로 조절되는 오토스케일링(Auto-scaling) 설정을 통해 비용 효율성을 극대화해야 한다. |

# System Context

## System Context Diagram

본 장에서는 '스마트 출입 솔루션'의 범위와 경계를 정의하고, 시스템이 상호작용하는 외부 사용자, 장치, 그리고 외부 시스템들을 명확히 기술합니다. 시스템 컨텍스트 다이어그램은 이러한 상호작용을 시각적으로 나타내며, 이를 통해 시스템이 전체 생태계 내에서 어떤 역할을 수행하는지 이해할 수 있습니다.

'스마트 출입 솔루션'은 단기 임대 숙소를 운영하는 호스트를 위해 반복적인 출입 관리 업무를 자동화하는 핵심 시스템입니다. 시스템은 다양한 외부 개체들과의 상호작용을 통해 이 목적을 달성합니다. 텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

## External Entity List

|  |  |  |
| --- | --- | --- |
| Name | Description | Related Stakeholder |
| 호스트 (Host) | [역할] 시스템의 핵심 관리자로서, 자신의 숙소와 스마트 도어락을 등록하고 예약 캘린더를 연동하여 출입 관리 자동화 규칙을 설정합니다.  [사용환경] 인터넷에 연결된 PC의 웹 브라우저 또는 모바일 기기(스마트폰, 태블릿)의 전용 앱을 통해 시스템에 접속합니다.  [전문성] 대부분 개인 숙소 운영자로, IT 기술에 대한 전문 지식은 없으나 에어비앤비와 같은 웹/앱 서비스 사용에는 익숙한 수준입니다. | 에어비앤비 개인 호스트 |
| 투숙객 (Guest) | [역할] 시스템이 생성한 임시 출입 코드를 전달받아 숙소를 이용하는 최종 사용자입니다.  [사용환경] 주로 스마트폰을 통해 SMS나 이메일로 출입 코드를 수신합니다. 여행 중인 경우가 많아 불안정한 네트워크 환경이나 기기 배터리 부족 상황에 놓일 수 있습니다.  [전문성] 기술 수준은 일반 사용자이며, 별도의 앱 설치나 복잡한 절차 없이 즉시 사용할 수 있는 간단한 출입 방식을 선호합니다. | 게스트 |
| 예약 플랫폼 | [역할] 에어비앤비, Vrbo 등 숙소 예약이 이루어지는 외부 플랫폼으로, 우리 시스템에 예약 데이터(체크인/체크아웃 시간)를 제공하는 원천 소스 역할을 합니다.  [SW 사양] 각 플랫폼은 iCalendar(.ics) 표준 형식의 캘린더 내보내기(Export) 기능을 API 형태로 제공합니다.  [품질 수준] 가용성: 시스템이 주기적으로 예약 정보를 폴링(Polling)해야 하므로 24/7 운영이 보장되어야 합니다. 정확성: iCal 피드에 포함된 예약 날짜 및 시간 정보의 정확성이 보장되어야 합니다. | 개발팀, 경영진 |
| 스마트 도어락 클라우드 플랫폼 | [역할] 물리적 스마트 도어락을 원격으로 제어하는 API를 제공하는 외부 시스템입니다. 우리 시스템은 이 플랫폼에 명령을 보내 기간 한정 임시 출입 코드를 생성 및 삭제합니다.  [SW 사양] Seam.co와 같은 통합 API 또는 Nuki, Schlage 등 개별 제조사의 클라우드 플랫폼이 해당됩니다. 대부분 OAuth 2.0 기반 인증과 REST API를 제공합니다.  [품질 수준] 신뢰성: API 호출 실패는 게스트의 출입 실패로 직결되므로 99.9% 이상의 높은 신뢰성이 요구됩니다. 성능: 출입 코드 생성 API의 응답 시간은 2초 이내여야 합니다. 보안: 모든 API 통신은 TLS로 암호화되어야 합니다. | 개발팀, 개인 호스트 |
| 알림 서비스 | [역할] 생성된 출입 코드나 시스템 오류 등 주요 정보를 호스트 및 투숙객에게 전달하는 메시징 전문 시스템입니다.  [SW 사양] AWS SES, Twilio 등 SMS, 이메일, 메신저 발송을 위한 API를 제공하는 외부 SaaS(Software-as-a-Service) 플랫폼입니다.  [품질 수준] 전달성: 메시지가 스팸으로 분류되지 않고 사용자에게 안정적으로 전달되어야 합니다. 성능: 특히 출입 코드와 같은 긴급 알림은 수 초 내에 지연 없이 발송되어야 합니다. | 운영팀, 개인 호스트 |
| 결제 시스템 | [역할] 유료 구독 플랜에 대한 결제를 안전하게 처리하는 외부 결제 게이트웨이입니다.  [SW 사양] Stripe, Toss Payments 등 PCI-DSS 인증을 받은 결제 전문 SaaS 플랫폼입니다.  [품질 수준] 보안: 신용카드 등 민감한 금융 정보를 다루므로 최고 수준의 보안이 요구됩니다. 가용성: 결제 시스템의 장애는 직접적인 매출 손실로 이어지므로 24/7 무중단 운영이 필수적입니다. | 경영진, 개인 호스트 |
| 호스트 (Host) | [역할] 시스템의 핵심 관리자로서, 자신의 숙소와 스마트 도어락을 등록하고 예약 캘린더를 연동하여 출입 관리 자동화 규칙을 설정합니다.  [사용환경] 인터넷에 연결된 PC의 웹 브라우저 또는 모바일 기기(스마트폰, 태블릿)의 전용 앱을 통해 시스템에 접속합니다.  [전문성] 대부분 개인 숙소 운영자로, IT 기술에 대한 전문 지식은 없으나 에어비앤비와 같은 웹/앱 서비스 사용에는 익숙한 수준입니다. | 에어비앤비 개인 호스트 |

## External Interface List

<작성 방법>

* System Context Diagram의 각 외부 인터페이스를 모두 나열한다.
* 시스템 관점에서 각 외부 인터페이스의 역할을 기술한다.
* 외부 인터페이스유형에 따라서 Interface 방법을 기술한다.
  + 사용자 인터페이스: 입출력 데이터
  + 디바이스 및 시스템 인터페이스: 입출력 데이터, 프로토콜, 데이터 포맷
* 품질(보안, 성능, 신뢰성 등)에 영향을 미칠 수 있는 interface의 특성(데이터 암호화, 인증방식, 입출력 크기, 주기/빈도, 등)을 기술한다.
* 인터페이스에서 발생할 수 있는 오류를 식별하고 오류 처리 방안에 대한 명세

<점검사항>

* 시스템과 외부 개체와의 인터페이스가 명확하고 구체적으로 표현되었는가?

|  |  |
| --- | --- |
| Name | Description |
| External Interface 1 | [역할]:  [입출력]:  [특성]:   * 프로토콜[선택적] * 데이터포맷[선택적] * 암호화 방식 * 인증 방식 * 입출력 크기 * 주기/빈도   [오류처리 방안] |
| External Interface 2 |  |
| External Interface 3 |  |
| External Interface 4 |  |
| External Interface 5 |  |

# Architectural Drivers

## Use Case Model

### Use Case Diagram

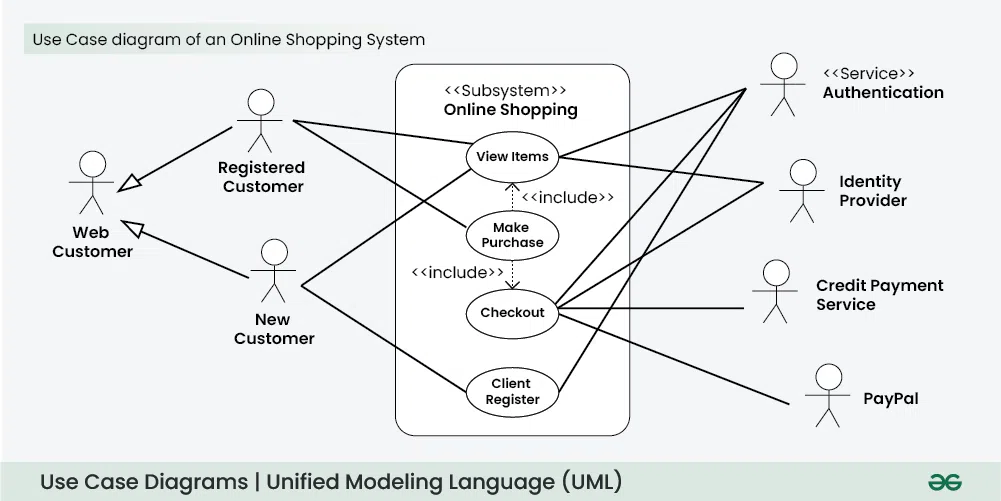
<작성 방법>

* 아키텍처 설계에서 고려할 핵심 기능을 모두 Use case로서 식별한다.
* 시스템과 상호작용하는 주요 대상을 Actor로 식별한다.
* Use case의 이름은 해당 Use case의 결과 즉 Actor에게 제공되는 결과물/서비스 등을 명확하게 기술해야 한다.
* Actor와 Use case 간의 연관 관계는 Actor와 Use Case 사이의 상호작용이 있을 때 표현한다.
* Use case 간의 포함 관계 (include)는 하나의 Use Case가 다른 Use Case를 필수적으로 포함하는 경우에, 확장 관계 (extend)는 하나의 Use Case가 특정 조건 하에서 다른 Use Case의 기능을 확장할 때만 사용해야 한다.
* Actor 간의 일반화 관계는 하나의 Actor가 다른 Actor의 추상화 혹은 일반화된 개념을 나타낼 /대 사용한다.

<점검사항>

* Use Case와 액터 사이의, Use Case 간의, 액터 간의 관계가 올바른가?

<예시>



### Actor List

<작성 방법>

* 시스템 컨텍스트 다이어그램의 외부 개체를 모두 Actor로 도출한다.
* Actor에 대한 설명이 시스템 관점에서 바라보았을 때의 Actor의 역할을 파악할 수 있도록 구체적이어야 한다.

|  |  |
| --- | --- |
| Name | Description |
| Actor1 |  |
| Actor2 |  |
| Actor3 |  |

### Use Case List

<작성 방법>

* Use Case Diagram에 표시된 모든 Use Case를 나열한다.
* Use case 이름은 해당 Use Case를 통해서 제공되는 시스템의 기능을 명확하고 구체적으로 표현해야 한다.
* Description은 시스템이 제공할 기능의 개요, BV 값 설정근거, AI 값 설정근거를 명확하게 기술해야 한다.
* BV (Business Value)는 해당 Use Case의 비즈니스 가치를 평가한다.
* AI (Architectural Impact)는 해당 Use Case가 아키텍처에 주는 영향도를 평가한다.
* Business Drivers은 해당 UC가 어떤 비즈니스 드라이버 (비즈니스 목표, 비즈니스 제약사항)으로부터 도출되었는지를 나타낸다.

<점검사항>

* 구조에 영향을 미치는 기능 명세가 충분한가?
* 유스케이스들의 우선순위(BV, AI)에 대한 평가가 타당한가?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name | Description | Priority | | Business Drivers |
| BV | AI |
| UC-01 |  | [개요]  [BV 설정근거]  [AI 설정 근거] |  |  |  |
| UC-02 |  |  |  |  |  |
| UC-03 |  |  |  |  |  |

\* BV: Business Value, AI: Architectural Impact << 최상 상 중 하 최하로 구분 >>

### UC-01 Title Description

<Pre Condition 작성 방법>

* Basic Flow 동작이 시작되기 위하여 필요한 시스템 상태를 Pre Condition으로 기술

<Post Condition 작성 방법>

* Basic Flow 동작이 종료되었을 때 충족해야 할 시스템 상태를 Post Condition으로 기술

<Basic Flow 작성 방법>

* 시스템이 제공할 기본 기능에 해당하는 동작만 기술한다.
* 각 스텝은 하나의 Actor 또는 시스템의 동작을 기술한다.
* 각 스텝은 Actor 또는 시스템을 주어로 하는 능동태 문장으로 기술한다.

<Alternative Flow 작성 방법>

* 시스템이 제공할 추가 기능 및 예외 상황 처리에 해당하는 동작을 기술한다.

<점검사항>

* 시나리오 명세가 구체적이고 명확한가?
* 아키텍처 측면에서 고려할 이슈를 포함하는 대안 시나리오가 식별되었는가?
* 해당 시나리오에 관련된 모든 Actor가 시나리오에서 언급되었는가?

|  |  |
| --- | --- |
| Pre Condition |  |
| Post Condition |  |
| Basic Flow |  |
| Alternative Flow |  |

### UC-02 Title Description

### *UC-03 Title* Description

## Quality Attribute Scenario

### QA Scenario List

<작성 방법>

* 비즈니스 목표 달성 위하여 필요한 모든 QA를 식별한다.
* Description은 품질 요구사항의 개요, BV 값 설정근거, AI 값 설정근거를 명확하게 기술해야 한다.
* BV (Business Value)는 해당 QA 시나리오의 비즈니스 가치를 평가한다.
* AI (Architectural Impact)는 해당 QA 시나리오가 아키텍처에 주는 영향도를 평가한다.
* Business Drivers은 해당 QA가 어떤 비즈니스 드라이버 (비즈니스 목표, 비즈니스 제약사항)으로부터 도출되었는 지를 나타낸다.

<점검사항>

* 비즈니스 목표를 실현하는데 필요한 품질요구사항이 충분한가?
* 품질요구사항의 우선순위(BV, AI)에 대한 평가가 타당한가?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Description | Type | Priority | | Business Drivers |
| BV | AI |
| QA-01 | [개요]  [BV 설정근거]  [AI 설정근거] |  |  |  |  |
| QA-02 |  |  |  |  |  |
|  |  |  |  |  |  |

\* 아키텍처 드라이버를 구성하는 QA 6개 이상 정의 (성능 QAS 1개 이상, 가용성 QAS 1개 이상, 변경용이성 QAS 1개 이상이 포함되어야 함)

### QA-01 Title Scenario

<작성 방법/점검사항>

* Description이 해당 QA Scenario에 대한 간략한 설명을 명확하게 기술하였는가?
* Source 및 Stimulus가 구체적인가? 즉 아키텍처 평가를 수행할 때 실제 어떤 입력이 사용되는 명확하게 판단할 수 있는가?
  + 나쁜 예) 기능의 추가
  + 좋은 예) 새로운 스케줄링 방법의 추가
* Artifact 및 Environment가 구체적인가? 즉 아키텍처 평가를 수행할 때 평가 대상인 시스템과 평가 환경을 명확하게 판단할 수 있는가? Environment는 Stimulus에 대한 Response를 관찰하며 Measure의 수치가 기대될 수 있는 구체적인 환경이 기술되어야 한다. 예) Stimulus 수신 당시 시스템의 부하
* Response가 구체적인가? 즉 해당 QA에 대한 평가를 수행할 때 Measure의 대상이 되는 동작/결과를 구체적으로 기술해야 한다.
* Response Measure가 Response와 일치하며 구체적인가? 즉 Measure의 유형 및 수치가 Response와 일치해야 하며 QA의 충족 여부 및 충족 수준을 판단할 수 있도록 Measure의 수치가 구체적인가?

|  |  |
| --- | --- |
| QA Type |  |
| Description |  |
| Source of Stimulus |  |
| Stimulus |  |
| Artifact |  |
| Environment |  |
| Response |  |
| Response Measure |  |

### QA-02 Title Scenario

## Architectural Constraint

<작성 방법>

* 앞서 도출한 비즈니스 제약 사항 중에서 아키텍처 설계 결정에 영향을 미칠 수 있는 요소를 모두 도출한다.

<점검사항>

* Title이 해당 architectural constraint의 핵심 사항을 명확하게 표현하고 있는가?
* Description이 해당 architectural constraint의 내용을 구체적으로 명확하게 기술하고 있는가?
* Business Drivers은 해당 architectural contraint가 어떤 비즈니스 드라이버 (비즈니스 목표, 비즈니스 제약사항)으로부터 도출되었는 지를 나타낸다.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Title | Description | Business Drivers |
| AC-01 |  |  |  |
| AC-02 |  |  |  |
|  |  |  |  |

# High Level Structure Design Description

## Domain Modeling

\* 우선순위가 높은 UC 3개 이상에 대해 작성함. Basic Flow 시나리오는 기본적으로 기술하고 아키텍처적으로 중요한 Alternative Flow 시나리오도 포함하여 기술해야 함.

### Conceptual Class List

|  |  |  |  |
| --- | --- | --- | --- |
| Conceptual Class | | | Relevant Use Cases |
| ID | Name | Type |
| CC-01 | UserData | entity | UC-01, UC-02 |
|  |  |  |  |

<작성 방법>

* Use case별로 해당 기능을 실현하기 위하여 필요한 Conceptual Class 목록을 도출한다.
  + Conceptual Class의 도출은 한번에 이루어기보다는 개별 Use Case를 기반으로 도출을 반복하면서 정제된다.
* Conceptual Class 카테고리에 따라서 entity, control, boundary, application logic 등의 스테레오타입 정보를 Type란에 기술한다.

<점검 사항>

* Use Case 시나리오를 실현하기 위해 필요한 개념적 클래스가 충분히 도출되었는가?
* 구현 수준의 클래스를 포함하고 있지 않은가?

### Dynamic View

#### *UC-01 Title* Use Case Dynamic Domain Model

<작성 방법>

* UC 시나리오를 UML Communication Diagram을 이용하여 conceptual class 인스턴스 간의 Message 흐름으로 표현한다.
* Messgae label은 **ReturnData := MessageName (argument-list)** 의 형식으로 표현한다.

<점검 사항>

* 인스턴스 간의 메시지가 메시지 형식에 맞춰서 작성되었는가?
* 인스턴스 간의 메시지 상호작용이 구체적이고 명확한가?
* 유스케이스 시나리오와 일관성이 있는가?

텍스트, 도표, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

#### *UC-02 Title* Use Case Dynamic Domain Model

### Static View

텍스트, 도표, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

<작성 방법>

* 도출된 conceptual class간의 association을 추가하여 클래스 다이어그램을 작성한다.
* Association에 multiplicity 정보를 추가한다.
* Entity 클래스에는 속성을 추가한다.

<점검 사항>

* Static View의 Class Diagram과 Dynamic View의 Communication Diagram간에 일관성이 있는가?

## Quality Driven Architectural Design

<작성 방법>

* 3장에서 식별된 각 QA를 위한 Architectural Design이 제시되어야 한다.
* 우선순위가 높은 QA 3개 이상에 대해 작성함.

### Exploring Architectural Options for *QA-01 Title*

#### Quality Driven Architectural Decision Structure



<작성 방법>

* 해당 QA를 만족시키기 위해서 설계 시 고려해야할 설계이슈(Design Issue)와 관련된 설계 옵션 (Design Option) 간의 의사결정 구조를 도식화 한다.
* 고려해야할 설계 이슈로는 다음과 같은 것을 고려해 볼 수 있다.
  + 컴포넌트 단위로 시스템 구조화 방안
  + 컴포넌트 제어 방안
  + 컴포넌트 간의 상호작용 방안
  + 데이터 접근 및 관리 방안
  + 하드웨어 리소스 관리 방안
  + 컴포넌트 배포 방안
* 해당 Design Issue가 다른 Design Option에서 추가적으로 파생된 문제인 경우에는 이들을 연결시켜서 표현한다.

<점검 사항>

* 품질 요구사항에 대한 아키텍처 설계 결정 구조가 적절한가?

#### Design Decisions for *설계 이슈1*

##### ***설계 이슈1***

*설계 이슈1*에 대한 설명 (배경 및 가정, 이슈 정의) 기술

|  |  |
| --- | --- |
| 배경 및 가정 |  |
| 이슈 정의 |  |

<작성 방법>

* 이슈가 발생하는 상황과 그 배경을 구체적으로 기술하고, 이슈와 관련된 가정된 조건이 있다면 이를 구체적으로 명시한다.
* 기술된 배경 및 가정하에서 제기된 이슈(문제)가 무엇인지 명확하게 정의하고, 이 문제가 왜 중요한지 설명한다.

<점검 사항>

* 설계 이슈가 다루는 문제가 구체적이고 명확한가?

##### **Design Options for *설계 이슈1***

|  |  |
| --- | --- |
| Design Options | |
| Title | Description |
|  |  |
|  |  |
|  |  |

<작성 방법>

* 해당 설계이슈와 관련된 모든 Design Option을 기술한다.
* Design Option의 Title이 설명하고 있는 Description과 일관성이 있어야 한다.
* Design Option의 Description이 실질적이고 세부적인 design이 제시되어야 한다.
  + 적용하고 있는 설계 솔루션(패턴, 택틱 등)에 대한 일반적인 설명이 아니라 해당 솔루션을 이 시스템에 적용할 때의 아키텍처 설계 결정을 구체적으로 제시해야 한다.
  + Layer patter n 적용의 경우: Layer의 수 및 각 layer의 역할, Layer interface 설계 등에 대한 decision이 제시될 필요가 있음
* 가능하면 제시된 설계 옵션들을 명확히 비교해 볼 수 있도록 구체적인 설계결정을 반영한 설계 뷰를 제시하는 것이 좋음

<점검 사항>

* 설계 옵션에 대한 설명이 구체적인가?

##### **Decision and Rationale for *설계 이슈1***

|  |  |  |  |
| --- | --- | --- | --- |
| Design Option Title | Pros | Cons | Rationale |
| 마이크로커널 스타일 적용  (selected) |  |  |  |
| 마이크로서비스 스타일 적용 |  |  |  |

<작성 방법>

* 각 Design Option별로 3장에서 제시된 모든 관련된 QA 측면 및 필요한 관심사(concerns) 측면에서 장/단점을 제시한다.
  + Pros/cons는 QA와 관심사 관점에서 구체적으로 장단점을 기술해야 한다. 즉 각 QA 관점에서 Response Measure에 대한 유/불리 와 Constraint 충족 여부 등이 명시적으로 제시될 필요가 있음
* 제시된 Design Option들 중에서 가장 적합한 Design Option을 선정하는 근거를 기술한다.

<점검 사항>

* 설계 옵션에 대한 평가와 선택 근거가 타당한가?

#### Design Decisions for *설계 이슈2*

#### Design Decisions for *설계 이슈3*

### Exploring Architectural Options for *QA-02 Title*

#### Quality Driven Architectural Decision Structure

#### Design Decisions for *설계 이슈1*

#### Design Decisions for *설계 이슈2*

## Component & Connector View

* 4.2절에서 기술된 아키텍처 설계 결정 위주로 표현한다.

|  |  |  |
| --- | --- | --- |
| QA Scenarios ID | (Selected) Design Option Title | Reflection in the C&C View |
| QA-01 | Design Option Title1 |  |
| Design Option Title3 |  |
| QA-02 |  |  |
|  |  |

<작성 방법>

* 4.2절에서 기술한 QA에 대한 아키텍처 설계결정 중에서 C&C View에 반영된 부분이 있으면 C&C View의 어느 부분에 어떻게 반영되어 표현되었는 지를 기술한다.

<점검 사항>

* 아키텍처 설계 결정이 구체적으로 올바로 반영되어 있는가?

### UML Component Diagram



### Component List

<작성 방법>

* Component Name은 컴포넌트가 제공하는 역할(responsibility)를 명확하게 나타낼 수 있도록 정의한다.
* Component Kind은 컴포넌트의 종류로서 다음과 같은 값을 가질 수 있다.
  + 분산 환경에서의 서브시스템 요소로서 client, server, data store
  + 동시 수행 요소로서 process (또는 task), thread
* Property Description에는 **해당 컴포넌트에 요구되는 기능 및 품질 요구사항, 제약사항 등** 아키텍처 드라이버 관점에서 기술한다.
  + 품질요구사항 및 제약사항에 대해서는 *완전만족/부분만족/미충족* 여부를 표시한다.
* Relevant ADs는 해당 컴포넌트와 관련된 모든 기능 및 품질요구사항, 제약사항을 나열한다.

<점검 사항>

* Property Description에 기술된 내용이 구체적이고 관련된 Architectural Drivers와의 일관성이 존재하는가?

|  |  |  |  |
| --- | --- | --- | --- |
| Component  Name | Component  Kind | Property Description | Relevant ADs |
| Component1 | process |  | UC-02  QA-01  QA-03 |
| … |  |  |  |

### Connector List

<작성 방법>

* Connector Name은 커넥터가 제공하는 컴포넌트 간의 상호작용을 명확하게 나타낼 수 있도록 명명한다. 잘 알려진 정의된 커넥터의 경우에는 해당 이름을 사용할 수 있다. (예, RPC, Message Queue, Message Broker 등)
* Property Description에는 해당 커넥터의 프로토콜(행위) 및 데이터 포맷 등이 어떠한 품질속성 및 기능과 관련 있는지를 기술한다.
* Relevant ADs는 해당 커넥터와 관련된 모든 기능 및 품질요구사항, 제약사항을 나열한다.

<점검 사항>

* Property Description에 기술된 내용이 구체적이고 관련된 Architectural Drivers와 일관성이 존재하는가?

|  |  |  |
| --- | --- | --- |
| Connector  Name | Property Description | Relevant ADs |
|  |  | UC-02  QA-01  QA-03 |
| … |  |  |

## Deployment View

* 4.2절에서 기술된 아키텍처 설계 결정 위주로 표현한다.

|  |  |  |
| --- | --- | --- |
| QA Scenarios ID | (Selected) Design Option Title | Reflection in the Deployment View |
| QA-01 | Design Option Title1 |  |
| Design Option Title3 |  |
| QA-02 |  |  |
|  |  |

<작성 방법>

* 4.2절에서 기술한 QA에 대한 아키텍처 설계결정 중에서 Deployment View에 반영된 부분이 있으면 어느 부분에 어떻게 반영되어 표현되었는 지를 기술한다.

<점검 사항>

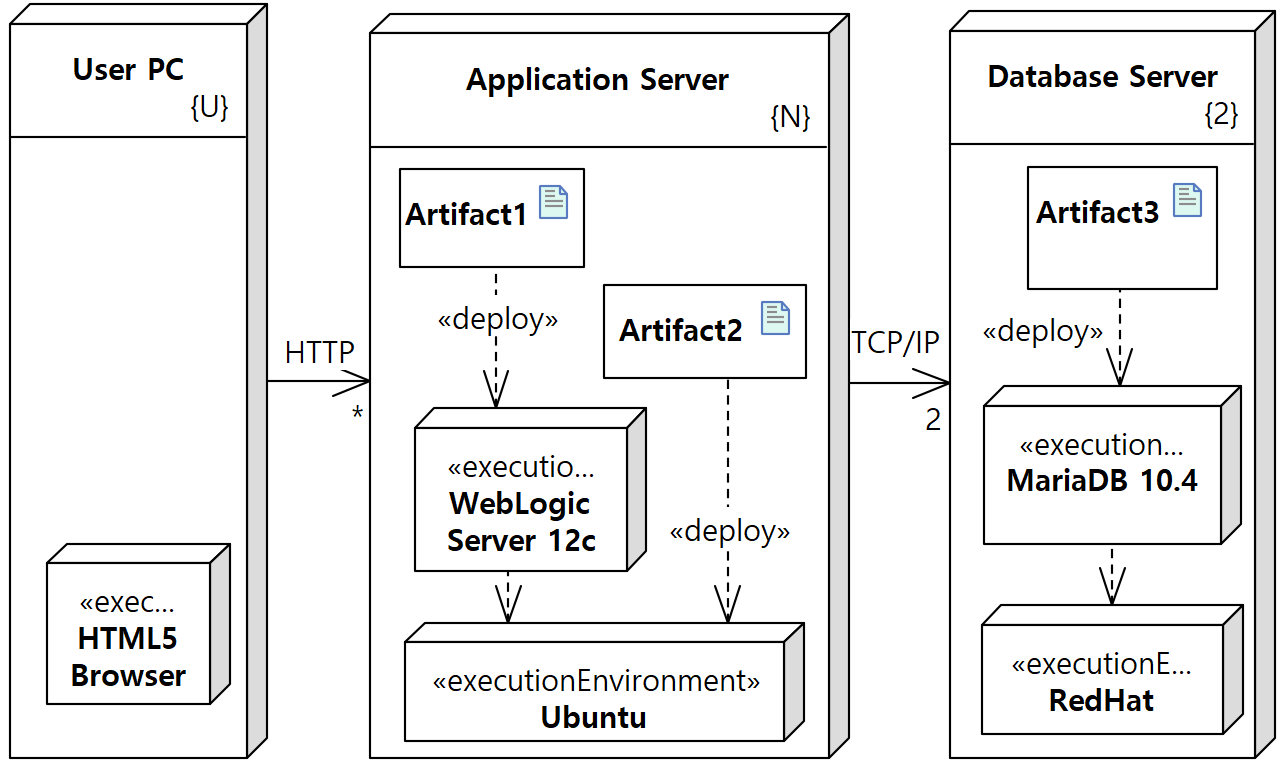
* 아키텍처 설계 결정이 구체적으로 Deployment View에 올바로 반영되어 있는가?

### Deployment Diagram

<작성 방법>

* 개발된 시스템 운용에 필요한 노드 및 노드 간의 통신과 노드의 실행 환경을 표현한다.
  + 노드의 유형과 역할(기능/QA)에 일치하는 구체적인 이름을 기술한다.
  + SW 컴포넌트의 실행에 필요한 SW 환경(플랫폼)을 기술할 필요가 있다면 <<execution environment>>로써 식별한다.
  + 각 노드 간의 필요한 모든 통신 경로를 식별하고 통신 방법을 구체적으로 기술한다.

<예시1>



#### Node Specification

<작성 방법>

* Deployment diagram에서 표시된 각 노드 별로 기술한다.
* 즉 각 노드가 시스템의 기능/QA 측면에서 어떤 역할을 하는지 설명이 필요하다. 복수 개 Multiplicity 인 경우 각 Instance의 역할을 명확히 기술한다. AD로 선택된 QAS의 만족을 보이기 위해 노드의 특성(CPU, Memory, HDD 등 HW 사양) 정보가 필요하다면 해당 정보를 기술한다.

<점검 사항>

* 각 노드의 설명이 구체적인가?

|  |  |
| --- | --- |
| Name | Description |
| User PC |  |
| Application Server |  |
| Database Server |  |

#### Execution Environment Specification

<작성 방법>

* Deployment diagram에서 <<execution environment>> 을 사용했다면 해당 항목별로 기술한다.
* Architectural driver(QA 등)의 만족을 위해 필요한 <<execution environment>> 특성(SW 제품 및 버전 등 SW 사양)과 결정 근거를 기술한다.

<점검 사항>

* 각 Execution Environment의 설명이 구체적인가?

|  |  |  |
| --- | --- | --- |
| Node | Name | Description |
|  | HTML5 Browser |  |
|  | WebLogic Server12c |  |
|  |  |  |

#### Communication Path Specification

<작성 방법>

* 각 통신 경로의 특성(통신을 위한 HW 및 SW 특성)과 그러한 특성을 결정한 근거를 3장에서 식별한 Architectural driver(QA 등)의 충족과 관련하여 정당화한다

<점검 사항>

* 각 통신 경로의 설명이 구체적인가?

|  |  |
| --- | --- |
| Path | Description |
| User PC – Application Server |  |
| Application Server – Database Server |  |

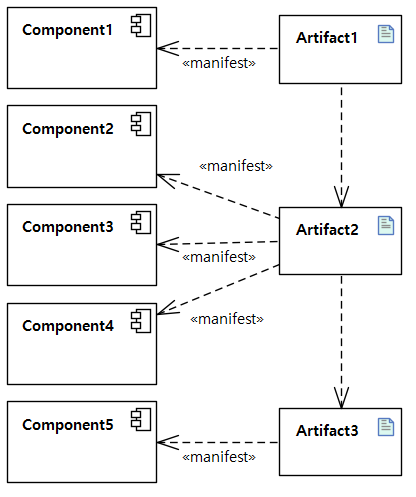
### Artifact Definition Diagram

<작성 방법>

* Deployment diagram의 Node에 설치될 각 파일들을 Artifact로서 식별한다.

<점검 사항>

* C&C View의 각 Component는 1개 이상의 Artifact에 포함되었는가?



#### Artifact Definition Specification

<작성 방법>

* Artifact definition diagram과 일치하여 Artifact, Manifested Components, Depends on을 기술한다.

<점검 사항>

* 각 Artifact의 기능/역할이 명확히 설명되었는가?.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Manifested Component | Depends on | Description and Rationale |
| Artifact1 | Component1 | Artifact2 |  |
| Artifact2 | Component2  Component3  Component4 | Artifact3 |  |
| Artifact3 | Component5 | None |  |

# Detailed Component Design Description

\* 아키텍처적으로 중요한 컴포넌트 3개 이상에 대해 작성하고, 컴포넌트의 대표적 Behavior를 시퀀스 다이어그램으로 기술해야 함.

## Design Description for *ComponentName*

### Overview



<작성 방법>

|  |  |
| --- | --- |
| 항목 | 설명 |
| 개요 | 이 컴포넌트의 역할을 한/두 줄로 간략하면서 명확하게 서술한다. |
| 컴포넌트 기능 요구사항 | 이 컴포넌트에 부여된 기능 요구사항을 서술한다. Provided Interface를 중심으로 구체적으로 설명한다. |
| 컴포넌트 품질 요구사항  및 제약사항 | 이 컴포넌트가 만족해야 할 품질 요구사항 및 제약사항을 서술한다.   * 시스템의 QA를 충족시키기 위하여 이 컴포넌트에서 만족시켜야하는 QA를 구체적으로 서술한다. * 시스템의 제약사항 중에서 이 컴포넌트에서 만족시켜야 하는 제약사항이 있다면 이를 구체적으로 서술한다. |

<점검 사항>

* 컴포넌트의 요구사항이 명확하게 정의되었는가?

### Quality Driven Component Design

#### Exploring Design Options for *QA-01 Title*

* + - * 1. **Quality Driven Component Design Structure**



<작성 방법>

* 해당 QA를 만족시키기 위해서 설계 시 고려해야할 설계이슈(Design Issue)와 관련된 설계 옵션 (Design Option) 간의 의사결정 구조를 도식화 한다.
* 고려해야할 설계 이슈로는 다음과 같은 것을 고려해 볼 수 있다.
  + 컴포넌트의 모듈 관점의 분해
  + 컴포넌트 간의 상호작용의 모듈 설계
* 해당 Design Issue가 다른 Design Option에서 추가적으로 파생된 문제인 경우에는 이들을 연결시켜서 표현한다.

<점검 사항>

* 품질 요구사항에 대한 컴포넌트 설계 결정 구조가 적절한가?

#### Design Decisions for *설계이슈1*

###### Definition for *설계이슈1*

설계 이슈(*설계이슈1*)에 대한 설명 기술

<작성 방법>

* 이슈가 발생하는 상황과 그 배경을 구체적으로 기술하고, 이슈와 관련된 가정된 조건이 있다면 이를 구체적으로 명시한다.
* 기술된 배경 및 가정하에서 제기된 이슈(문제)가 무엇인지 명확하게 정의하고, 이 문제가 왜 중요한지 설명한다.

###### Design Options for *설계이슈1*

|  |  |
| --- | --- |
| Design Option | |
| Title | Description |
|  |  |
|  |  |
|  |  |

<작성 방법>

* 해당 설계이슈와 관련된 모든 Design Option을 기술한다.
* Design Option의 Title이 설명하고 있는 Description과 일관성이 있어야 한다.
* Design Option의 Description이 실질적이고 세부적인 design이 제시되어야 한다.
  + 적용하고 있는 설계 솔루션(패턴, 택틱 등)에 대한 일반적인 설명이 아니라 해당 솔루션을 이 시스템에 적용할 때의 아키텍처 설계 결정을 구체적으로 제시해야 한다.
  + Layer patter n 적용의 경우: Layer의 수 및 각 layer의 역할, Layer interface 설계 등에 대한 decision이 제시될 필요가 있음
* 가능하면 제시된 설계 옵션들을 명확히 비교해 볼 수 있도록 구체적인 설계결정을 반영한 설계 뷰를 제시하는 것이 좋음

<점검 사항>

* 설계 이슈와 설계 옵션에 대한 설명이 구체적인가?

##### Decision and Rationale for *설계이슈1*

|  |  |  |  |
| --- | --- | --- | --- |
| Design Option Title | Pros | Cons | Rationale |
| Design Option Title1 |  |  |  |
| Design Option Title2 |  |  |  |

<작성 방법>

* 각 Design Option별로 5.1절에서 제시된 모든 관련된 QA 측면 및 필요한 관심사(concerns) 측면에서 장/단점을 제시한다.
  + Pros/cons는 QA와 관심사 관점에서 구체적으로 장단점을 기술해야 한다. 즉 각 QA 관점에서 Response Measure에 대한 유/불리 와 Constraint 충족 여부 등이 명시적으로 제시될 필요가 있음
* 제시된 Design Option들 중에서 가장 적합한 Design Option을 선정하는 근거를 기술한다.

<점검 사항>

* 설계 옵션에 대한 평가와 선택 근거가 타당한가?

#### Exploring Design Options for *QA-02 Title*

### Module View

* 5.1.2절에서 기술된 컴포넌트 설계 결정 위주로 표현한다.

|  |  |  |
| --- | --- | --- |
| QA Scenarios ID | (Selected) Design Option Title | Reflection in the Module View |
| QA-01 | Design Option Title1 |  |
| Design Option Title3 |  |
| QA-02 |  |  |
|  |  |

<작성 방법>

* 4.2절에서 기술한 QA에 대한 아키텍처 설계결정 중에서 Module View에 반영된 부분이 있으면 어느 부분에 어떻게 반영되어 표현되었는 지를 기술한다.

<점검 사항>

* 컴포넌트 설계 결정이 모듈 뷰에 구체적으로 반영되어 있는가?

#### Class Diagram

<작성 방법>

* Component를 구현하기 위하여 필요한 Class 및 Interface들을 모두 식별한다.
* Class 간의 관계(generalization, association, composition/aggregation, dependency)를 표시한다.
* Class의 attribute 및 operation이 구체적으로 명시한다.
* 많은 수의 Class 및 Interface가 식별된 경우 Package Cohesion/Coupling을 고려하여 Packaging한다.

<점검 사항>

* Class Diagram이 구체적이고 올바로 작성되어 있는가?



##### Element List

<작성 방법>

* 위의 Class diagram에 표현된 모든 요소(package, class, interface)에 대한 역할을 설명한다.
* 각 요소가 Component에 부여된 요구사항(기능, 품질, 제약사항 등)에 어떻게 기여하는 지 측면에서 설명한다.

<점검 사항>

* Element의 설명이 구체적인가?

|  |  |
| --- | --- |
| Name | Description |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

#### Sequence Diagram

<작성 방법>

* 컴포넌트가 제공하는 인터페이스의 각 오퍼레이션이 어떻게 내부 모듈 인스턴스 간의 상호작용으로 실현되는가를 시퀀스 다이어그램을 이용해서 기술하시기 바랍니다.
* 시작은 컴포넌트 provided IF가 커넥터 모듈로부터 호출되는 방식으로 기술하시기 바랍니다.

<점검 사항>

* Sequence Diagram이 구체적이고 올바로 작성되어 있는가?



## Design Description for *Component 2*

## Design Description for *Component 3*

# Architectural Evaluation

* 우선순위가 높은 QA 3개 이상에 대해 이와 선정된 Architectural Decision에 대한 분석서를 작성한다.

## Architectural Evaluation for *QA-01 Title*

<작성 방법>

* 제시된 양식에 맞춰서 작성한다.

<점검 사항>

* 아키텍처 결정의 민감점, 절충점, 위험요소, 비위험요소 분석이 타당한가?
* 아키텍처 결정에 대한 Reasoning이 타당한가?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| QA 시나리오 |  | | | |
| 품질속성 |  | | | |
| 자극 소스 |  | | | |
| 자극 |  | | | |
| 환경 |  | | | |
| 응답 |  | | | |
| 측정 |  | | | |
| 아키텍처 결정 | 위험요소 | 민감점 | 절충점 | 비위험요소 |
|  | R1 | S1 | T1 |  |
|  |  |  |  | NR1 |
|  |  |  |  |  |
| Reasoning | <<아키텍처 결정들이 시나리오가 표현하는 품질속성을 달성하는데 어떻게 공헌하는지에 대한 근거를 설명>> | | | |

### List of Risks

* R1
* …

### List of Sensitivities

* S1
* …

### List of Tradeoffs

* T1
* …

### List of Nonrisks

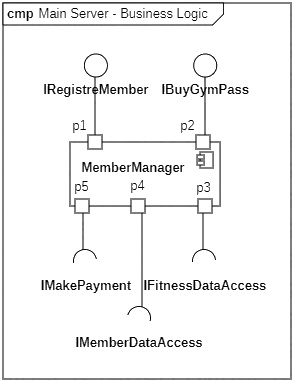
* NR1
* …

Appendix

## Detailed Component Specification for C&C View

### Component 1 Name

<< 아래 그림과 유사하게 UML 컴포넌트 다이어그램으로 컴포넌트의 인터페이스를 모델링하기 바랍니다.>>



##### Interface List

<점검 사항>

* 해당 Component의 모든 interface가 정의되었는가?
* 각 interface의 유형(Provided/Required)가 UML Component Diagram과 부합하는가?
* 각 interface의 역할/기능을 명확하게 설명하였는가?

|  |  |  |
| --- | --- | --- |
| Interface Name | Kind | Description |
| IRegisterMember | Provided | 회원 가입을 요청하는 interface |
| IBuyGymPass | Provided | 이용권 구매를 요청하는 interface |
| IMakePayment | Required | 결제를 요청하는 interface |
| IMemberDataAccess | Required | DB에 저장된 회원 정보에 접근하는 interface |
| IFitnessDataAccess | Required | DB에 저장된 피트니스 정보에 접근하는 interface |

### Component 2 Name

## Interface specifications for C&C View

### *Interface2\_1* Interface Specification

<작성 방법>

* 인터페이스를 구성하는 각 operation 별로 구체적인 인자/반환 타입을 기술한다.

<점검 사항>

* Component에 할당된 모든 기능이 Operation으로 할당되었는가?
* Operation의 이름이 제공되는 기능을 명확히 나타내는가?
* Operation의 인자/반환 타입이 올바르게 정의되었는가?

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| Operation | Responsibility |
| op1() |  |
| op2() |  |