1.1.1 Statkeholders List

<작성 방법>

* 개발하고자 하는 제품에 대한 획득, 개발, 인증, 배포/판매, 운영/관리, 유지보수 등 다양한 이해관계자(조직)를 식별한다.
* 이해관계자의 역할, 관심사/애로사항(concern) 등의 배경을 기술한다.

1.1.2 Business Goals

<작성 방법>

* 이해관계자의 관심사항을 반영하여, 조직이 달성하고자 하는 구체적이고 측정가능한 비즈니스 가치를 정의한다.
  + 경영자: 시장점유율 확대, 매출 확대
  + 고객: 고객 만족도 개선, 제품 및 서비스의 품질 향상
  + 투자자: 수익성 확대
  + 공급자: 공급망의 효율성 및 지속가능성 향상
  + 지역사회 및 환경: 사회적 책임 및 환경 지속 가능성 강화
  + 정부 및 규제기관: 규제 준수 및 좋은 기업 시민 되기
* 비즈니스 목표와 관련된 아키텍처와의 연관성을 기술한다.
* 비즈니스 목표의 중요도를 조직의 전략적 방향과 현재 상황에 따라서 결정한다.

<점검사항>

* 비즈니스 목표가 구체적이고 측정가능한 비즈니스 가치를 반영하였는가?
* 비즈니스 목표의 중요도에 대한 판단 근거가 합당한가?

1.2 Business Constraints

<작성방법>

* 예산, 시간, 법적, 자원 제약 등을 포함하여 구체적이고 명확하기 기술한다.
* 비즈니스 제약사항의 아키텍처에 대한 영향도를 기술한다.

<점검사항>

* 비즈니스 제약사항 서술이 구체적이고 명확한가?
* 비즈니스 제약사항의 아키텍처에 대한 영향도 서술이 타당한가?

2.1 System Context Diagram

<작성 방법>

* 개발 대상 시스템 자체와 외부 개체(External Entity)을 명확하게 구분되어야 한다.
* 시스템과 상호작용하는 외부개체가 1개 이상인 경우에는 이에 대한 표현이 필요함 (노트로 표현 가능). 또한 시스템과 상호작용하는 외부개체의 개수에 대한 가정 및 근거를 기술하는 것이 필요합니다.
* 개발 대상 시스템과 외부 개체와의 인터페이스를 Provided/Required Interface 관점에서 표현한다
* 사용자와 개발 대상 시스템과의 사용자 인터페이스는 화면별로 별도의 인터페이스로 표시한다.

<점검사항>

* 시스템의 경계(정의)가 명확한가?
* 시스템과 직접적으로 상호 작용하는 모든 대상(사용자, 시스템, 장치 등)이 외부 개체로서 식별되었는가?

2.2 External Entity List

<작성 방법>

* 비즈니스 목표와 관련된 이해관계자 중에서 시스템과 직접적인 상호작용을 하는 대상은 모두 외부개체로 도출한다.
  + 이해관계자가 직접 상호작용하지 않고 이해관계자가 운영하는 시스템에 의해서 상호작용하는 경우에는 해당 시스템을 외부개체로 도출한다.
* 각각의 외부개체에 대해서 시스템 관점에서의 역할 및 특성을 기술한다.
  + 사용자 유형의 외부개체 특성으로, 사용자의 사용환경, 전문성 및 기술수준 등을 구체적으로 기술한다.
  + 사용자 유형 이외의 외부개체 특성으로, 시스템 및 장치의 HW 및 SW 사양, 품질 수준(성능, 신뢰성, 가용성 등)을 기술한다.

<점검사항>

* 외부 개체의 역할과 특성이 명확하고 구체적으로 기술되었는가?

2.3 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?

3.1.1 UseCase 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?

3.1.2 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.

3.1.3 UseCase 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?

3.1.x UC-xx 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?

3.2.1 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?

3.2.x QA-01 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?

3.3 Architectural Constraints

<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.