Title:
Architecture Description by HP

Date: 2021-06-14
Team 4

김상권, 이태영, 배재욱

Version	Document maturity (draft / valid)	Date of Issue (20xx-MM-DD)	Author/Owner	Check/Release	Description
1.0	draft	2021-06-14	김상권, 이태영, 배재욱		Initial version

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. System Purpose

I.1 Purpose

- 분산 자판기 시스템의 자판기 controller SW 개발

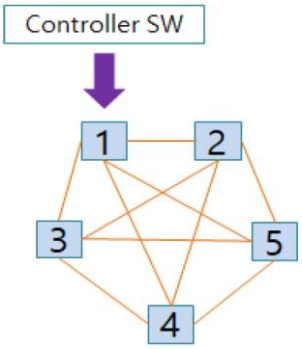

Figure 1. 분산 자판기 시스템 다이어그램

I.2 Context

I.2.1 Stakeholders

분산 자판기 시스템 개발에 앞서 Stakeholder 에 따른 주요 역할 및 희망사항에 대해 설명한다.

Table 1. Stakeholder List

Stakeholders	주요역할	희망사항-Goal	관련 QA
사장	사업 총괄 / 해당 사업의 시장성, 기술적 측면, 사업전개능력, 예상 경쟁력, 경제성 등을 판단하여 사업의 진행방향을 결정한다.	- 기존 코드 중 재활용 가능한 부분의 활용을 통한 낮은 개발비용/ 빠른 개발속도 달성 - 시장의 수요가 변화할 때 개발 제품의 수정이 용이해야 한다.	(1) Marketability (2) Modifiability
마케팅 담당자	시장 분석, 시장의 요구도 도출 및 이를 기반으로 한 제품 기획 / 영업 및 홍보를 통해 산출된 제품 판매	- 다양한 모델의 자판기에 적용 가능해야 한다. - 후속지원이 용이해야 한다. - SW 의 기능이 분명하고 간결하며 사용이 편리해야 한다.	(1) Portability (2) Manageability (3) Usability
사용자	시스템의 이해와 사용 방법을 숙지하고, 해당 서비스를 이용한다.	- 모든 구매한 물품과 비용이 한 눈에 보일 수 있게 해야 한다. - 희망 음료를 판매하는 가장 가까운 자판기를 안내 - 인증 코드는 범용성 있는 것을 사용했으면 한다.	(1) Usability (2) Usability (3) Usability
앱 개발자	사용자에게 서비스를 제공하기 위해 시스템 개발한다.	- 자판기 작동 시 일주일 정도는 이상 없이 동작해야 한다. - 예외처리 케이스 발생 시에도 자판기가 동작되어 함. - 화면 사이의 연결이나 연동이 사용자로 하여금 불편함을 느끼지 않도록 0.5 초 이내로 제한한다. - 사용자가 쓴 금액이 손실되지 않도록 시스템이 동작해야 한다.	(1) Availability (2) Reliability (3) Performance (4) Reliability
네트워크 개발자	네트워크 시스템을 주어진 요구사항에 따라 설계하고 구현한다.	- 네트워크가 끊긴 경우에도 로컬에서 동작 가능 - 네트워크 프로토콜은 간단하여 수정하기 쉬웠으면 한다. 여러 이용자가 동시에 사용해도 네트워크는 크게 영향 받지 않았으면 한다.	(1) Reliability (2) Modifiability (3) Scalability
유지보수 담당자	시스템 개발 완료 이후, 운영 중 기능/품질 문제 발생 시 문제 해결하고 관리한다.	- 프로그램 문제 발생 시 수정이 용이했으면 좋겠다. - 음료 메뉴 추가나 삭제가 용이했으면 좋겠다. - 기기 고장으로 장비 교체할 경우, 초기화 과정이 간단했으면 한다.	(1) Modifiability (2) Modifiability (3) Modifiability

I.2.2 Quality attributes (System Properties Web)

1) Modifiability

- 높은 가성비로 시스템을 신속하게 변경할 수 있는 능력

2) Usability

- 물건, 서비스를 어떤 특정 목적을 달성하기 위해 사용할 때에 얼마나 쉽게 사용할 수 있는가를 말합니다.
- 사용자로 하여금 쉽고, 편리하게 음료를 구매 할 수 있도록 분산자판기기 시스템을 개발해야 한다.

3) Reliability

- 시간이 지나도 계속 작동 가능한 시스템의 능력

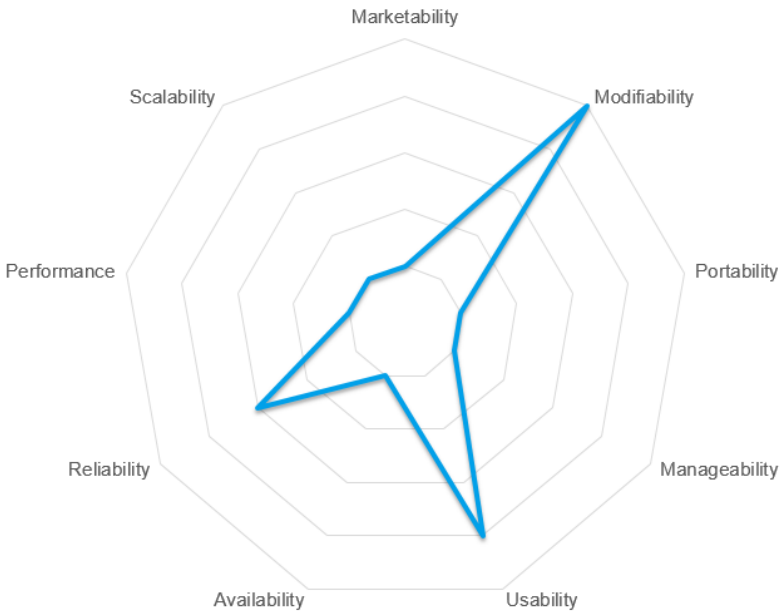

Figure 2. 선정된 Quality attribute(QA) 그래프

I.2.3 Scenario Brainstorming

- Web 항목 기준 각 5분 Brainstorming 한 결과 18개의 Raw Scenario 생성하였다. 각각의 Stakeholder 들의 의견을 반영하여 시나리오에 대한 Top Quality attribute 를 선정하였다. 그 결과 아래 빨간 박스 안 6가지의 Quality attribute 를 확인할 수 있었다.

Figure 3. Top Quality Attribute

Table 2. QA(Quality Attribute) Scenario Refinement

N	Quality	Source	Stimulus	Artifact	Environment	Response	Response Measure
1	Marketability	External	기존 코드 및 구조 재사용	System	Design time	Capture result	아키텍처 패턴 30% 이상 변경 적용에 비용 10% 미만
2	Performance	App developer	User Interface 간의 화면 전환	User interface unit	Development time	Capture result	화면 전환 시간 측정 0.5 초 이내(목표 0.5 초 이내 수준)
3	Modifiability	Unit testers	- Menu table changes - Specific menu removal	User interface unit	Run	Capture result	Menu changes complete in one hour
4	Usability	User	사용법 습득	System	Run	Capture result	5 분 안에 처음 사용자가 모든 기능 사용법 습득
5	Usability	User	- 구매할 물품 선택 - 구매할 물품 보기 Function 동작	User Interface unit	Run	Capture result	구매한 물품 리스트, 각 물품에 대한 비용, 총 비용을 모두 한 화면에 출력
6	Availability	System tester	- 시스템 전원 인가 후 각 기능을 반복적으로 동작	System	Run	Capture result	동작 시간 측정하여 168 시간 이내(일주일 기준)
7	Usability	User	- 희망 음료 찾기 실행 - 주변 자판기의 희망 음료 재고정보 얻기	Code unit	Run	Capture result	5 초 이내로 희망 음료 재고가 있는 자판기 중 최단 거리의 자판기 추천
8	Usability	User	- 타 자판기 희망 음료를 선 결재 후 인증코드 받기	Code unit	Run	Capture result	인증코드 받을 때, 인증코드 생성 및 응답 시간 2 초 이내
9	Portability	External	- 다양한 기존 자판기 모델에 적용	System	Design time	Capture result	3 종 이상의 자판기에 적용 가능할 것
10	Manageability	External	- 작동 현황의 모니터링	System	Run	Capture result	30 분 안에 원격으로 작동 상태에 대한 정보 획득
11	Scalability	System testers	- All vending machines operate their one cycle process at the same time continuously	Communication unit	Run	Capture result	Less 120% process time than a single vending machine process

I.3 System Interface

Iteration Goal 인 초기 전반적인 시스템 구조 결정을 위해 DVM 시스템을 Elements 로 선택하였다.

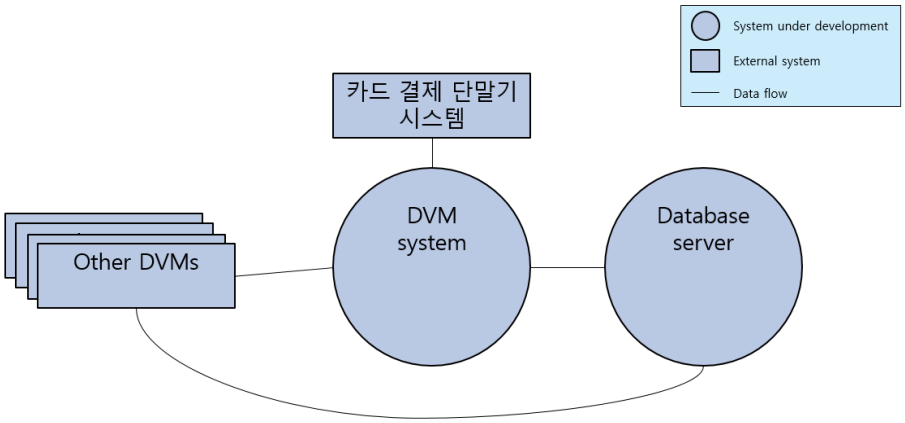

Figure 4. 전체 DVM 시스템 구조

I.4 Non-Functional Requirements

Table 3. Constraints & Concerns

ID	Constraints
CON-1	각 자판기는 모두 네트워크에 연결되어 있고 네트워크 연결 정보는 미리 알고 있다(최대 10 대).
CON-2	자판기의 판매 음료 종류는 사전에 결정된다.
CON-3	자판기 사이의 msg protocol 은 사전에 결정된다.

ID	Concern
CRN-1	처음에 전반적인 system architecture 부터 설계한다.
CRN-2	개발 환경은 프로그램 개발자와 네트워크 개발자가 익숙한 것을 선택한다.
CRN-3	모든 Architectural Driver 를 만족하는 Software Architecture 를 세운다.
CRN-4	사용자에게 적합한 UI/UX design 개발
CRN-5	사용자의 요구를 분석하여 그것들을 컴퓨터에 저장할 수 있는 데이터베이스의 구조에 맞게 변형한 후 특정 DBMS 로 데이터베이스를 구현
CRN-6	웹 방화벽 구축, 침입탐지시스템 등 정보보호를 위한 관리적 기술적 물리적인 시스템 구축

II. Structure

II.1 Overview

II.1.1 Design Concepts

Design Concepts 선정에 앞서 개발하려는 DVM 시스템의 구조적 이해뿐만 아니라, Reference Architecture, Deployment Pattern 종류와 각각의 개념 및 이해가 필요하다. Logical structure 를 표현하는 Reference Architecture 종류에는 Web Application, Rich Client Applications, Rich Internet Application, Mobile Application, Service Application 이 있고, DVM 시스템 구조에 적합한 Rich Client Application Reference Architecture 와 Service Application Reference Architecture 를 선정하게 되었다. Physical structure 를 표현하는 Deployment Pattern 종류에는 Two-Tier(Client-Server), Three-Tier, Four-Tie 가 있고, 우리 시스템에 적합한 Two-Tier Deployment Pattern 을 선정하였다.

II.1.1.1 Initiate Reference Architecture

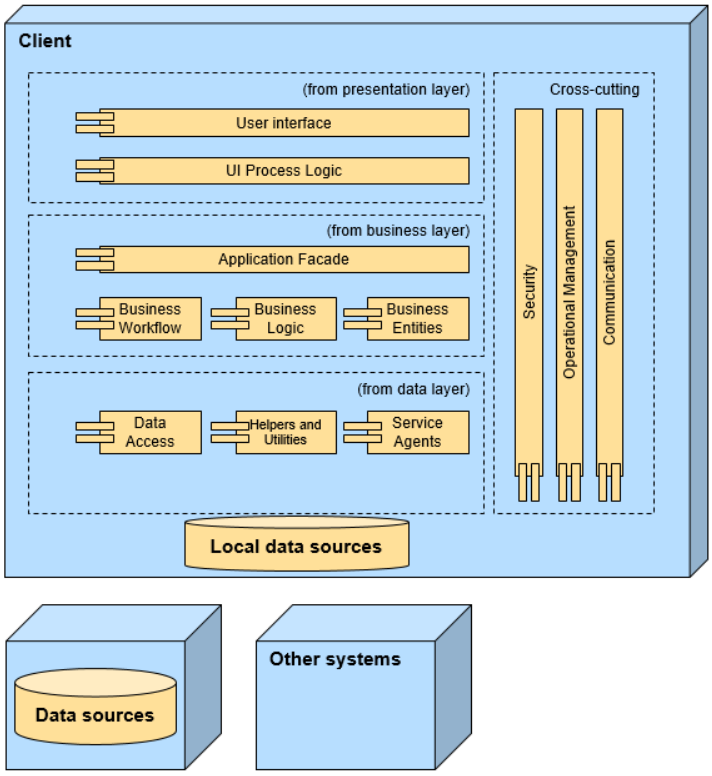

Figure 5. Rich client application reference architecture

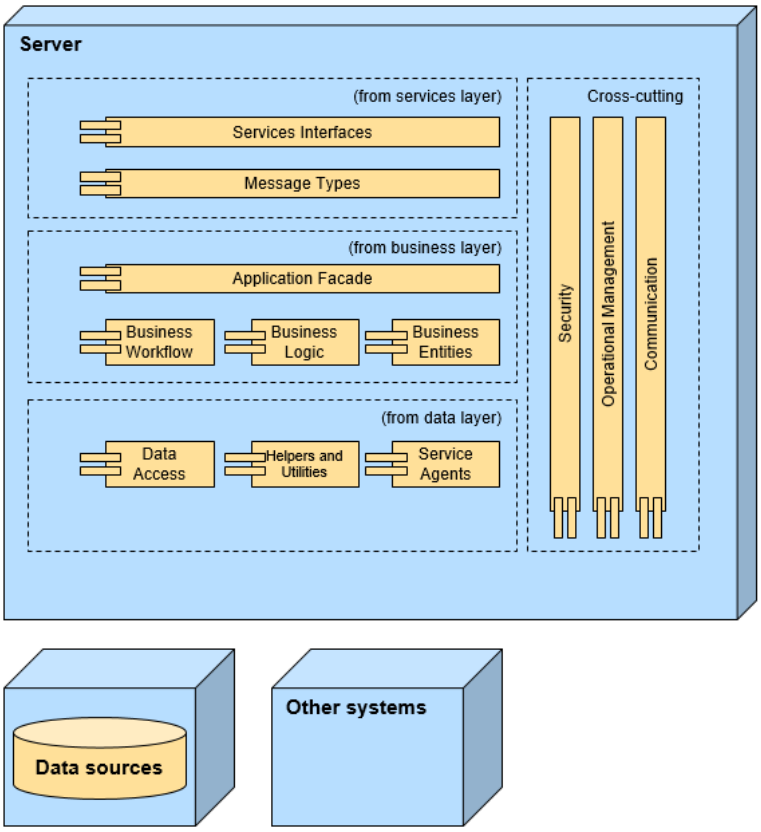

Figure 6. Service client application reference architecture

Design Decisions and Location	Rationale
Logically structure the client part using the Rich Client Application reference architecture	DVM system 은 client part 역할이며, UC-2, UC-3, UC-4, UC-5 처럼 대부분 기능 요구사항을 해결할 수 있는 능력이 필요함. 따라서 수행 capability 가 큰 rich client application 을 선택
	Discarded alternatives: web application, rich internet application 을 고려하였으나, DVM 요구사항에는 인터넷이 필요하지 않기 때문에 선정하지 않음
Logically structure the server part of the system using the Service Application reference architecture	UI 제공이 필요하지 않고, UC-1, QA-6 를 만족시키기 위해 데이터베이스 관리를 할 수 있는 application 필요
	Discarded alternatives: No discarded alternatives

II.1.1.2 Deployment Pattern

Design Decisions and Location	Rationale
Physically structure the application using the Two-tier Deployment pattern	QA-1, 2 을 고려하여 배포할 시스템의 Architecture 는 간단할 수록 좋고, 각각의 DVM 은 Web application 을 거치지 않고 직접 database server 에 접근하므로 two-tier deployment pattern 이 적절하다고 판단
	Discarded alternatives: No discarded alternatives

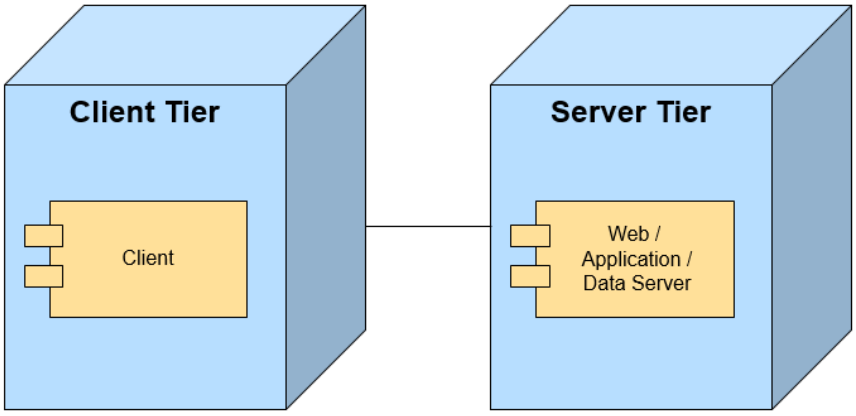

Figure 7. Two-tier Deployment pattern

II.1.1.3 Selected & Improved Reference Architecture

Design Decision and Location	Rationale
Remove **local data sources** in the rich client application	It is believed that there is no need to store data locally, as the network connection is generally reliable. also, communication with the server is handled in the data layer. internal communication between components in the client is managed through local method calls and does not need particular support
Create a module dedicated to **accessing the database servers** of DVM stock in the data layer of the Service Application reference architecture.	The service agents component from the reference architecture is adapted to abstract the access to the database servers of DVM stock. this will play a critical role in the achievement of UC-4 and UC-5. as shown in CON-1, all vending machines are connected to the network, and you need to know the network connection information.

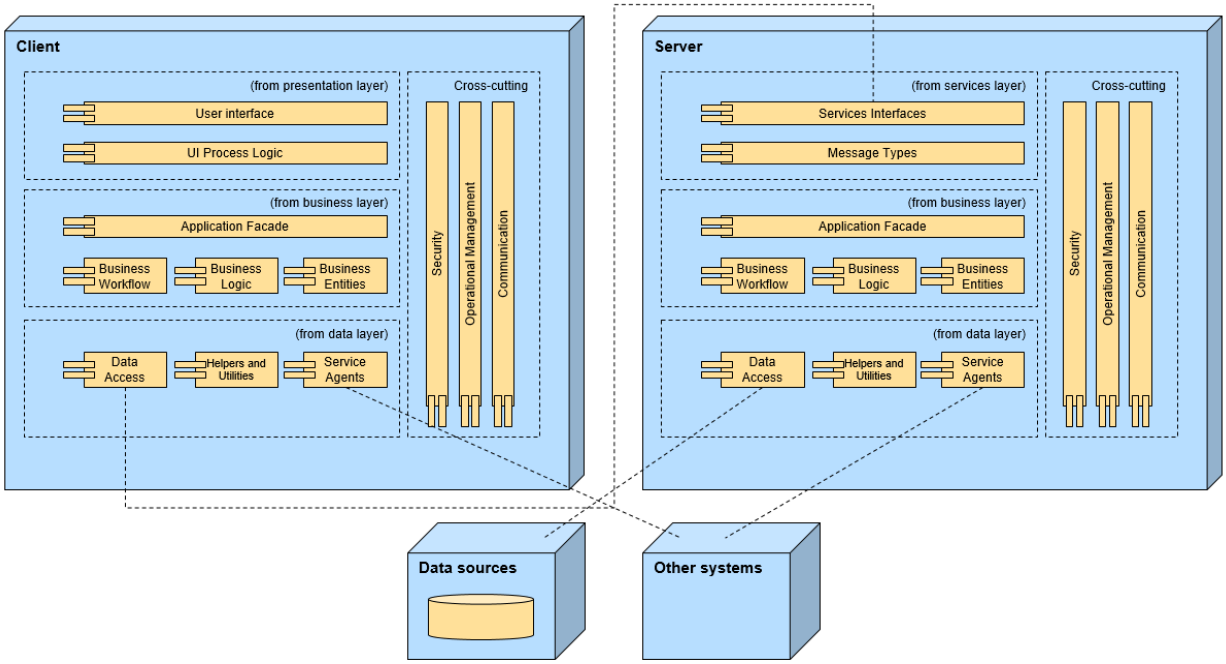

Figure 8. Total Selected Reference Architecture

Availability 를 중요한 QA 로 선정 후 선택한 Reference Architecture 토대로 Refinement 추가적으로 진행하였다. Availability QA 고려를 통해 Physical node 를 Refine 하기로 결정하였고, Two-tier deployment model 서버 tier 의 Application & Data server 를 선택하였다. (QA-5: Availability 자판기 작동 시 일주일 정도는 이상 없이 동작해야 한다.)

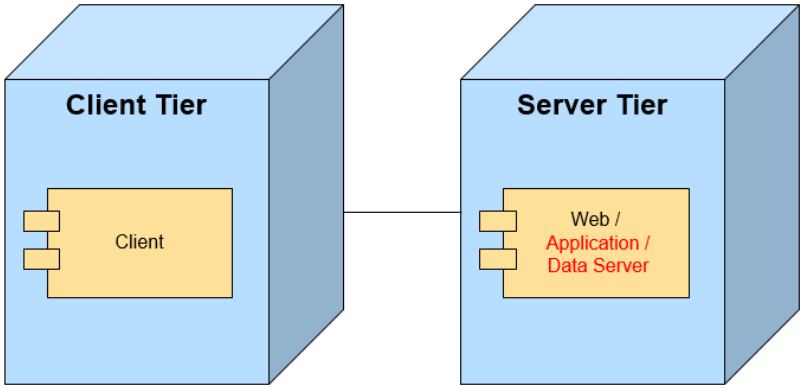

Figure 9. Selected Deployment pattern

Design Decision and Location	Rationale
Apply the Active redundancy tactic by refining the application server and other critical components such as the network management	중요한 elements 에 대해 시스템 redundancy 를 높일 수 있는 전략을 추가(load balancing)
Introduce an element from the **message queue technology** family	Queue 구조의 elements 를 시스템 failure 발생 시 치명적인 부분에 추가하여 trapping 하고자 하며, 이는 QA-5 Availability 를 고려한 사항
React native framework	Portable local user interface 를 구축하기 위한 JavaScript 언어의 framework 를 기반으로 개발

II.2 Components

II.2.1 Overview

Design Decision and Location	Rationale
Create a **Domain model**	Primary use cases 와 관련된 초기 domain model 을 생성하여 디자인 단계를 가속화 한다.
Identify **Domain Objects** that map to functional requirements	도메인 개체의 초기 식별은 시스템의 use cases 를 분석하여 만들 수 있습니다.
Decompose Domain Objects into general and specialized Components by layer-specific modules with an explicit interface	모든 Primary use cases 에 대해 앞서 식별해둔 domain objects 를 세분화하고 연결하기 위해 layer-specific modules 을 식별한다.

II.2.2 Initial Domain Model

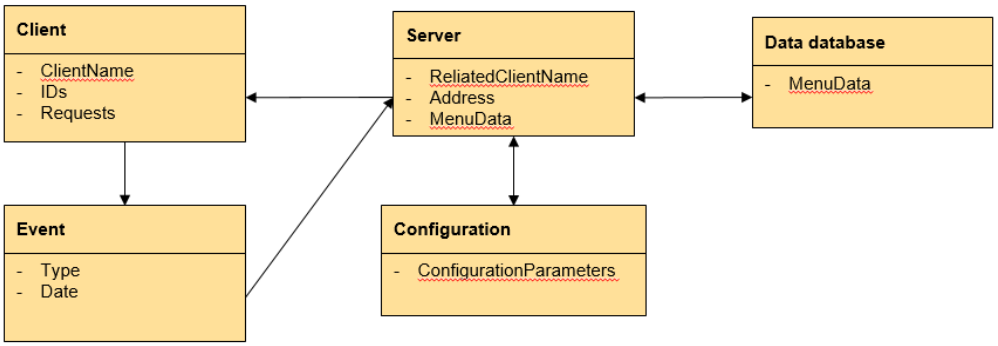

Figure 10. Initial Domain Model

II.2.3 Domain Objects Associated with Use Cases

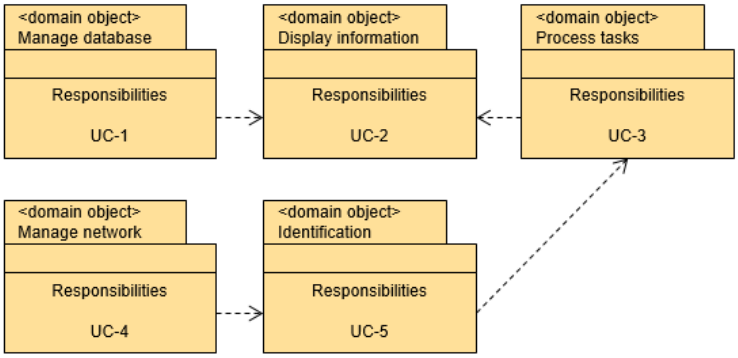

Figure 11. Domain Objects Associated with Use Cases

Design Decision and Location	Rationale
Apply the Active redundancy tactic by refining the application server and other critical components such as the network management	중요한 elements 에 대해 시스템 redundancy 를 높일 수 있는 전략을 추가(load balancing)
Introduce an element from the **message queue technology** family	Queue 구조의 elements 를 시스템 failure 발생 시 치명적인 부분에 추가하여 trapping 하고자 하며, 이는 QA-5 Availability 를 고려한 사항
React native framework	Portable local user interface 를 구축하기 위한 JavaScript 언어의 framework 를 기반으로 개발

II.3 Interface

II.3.1 Interface Structure

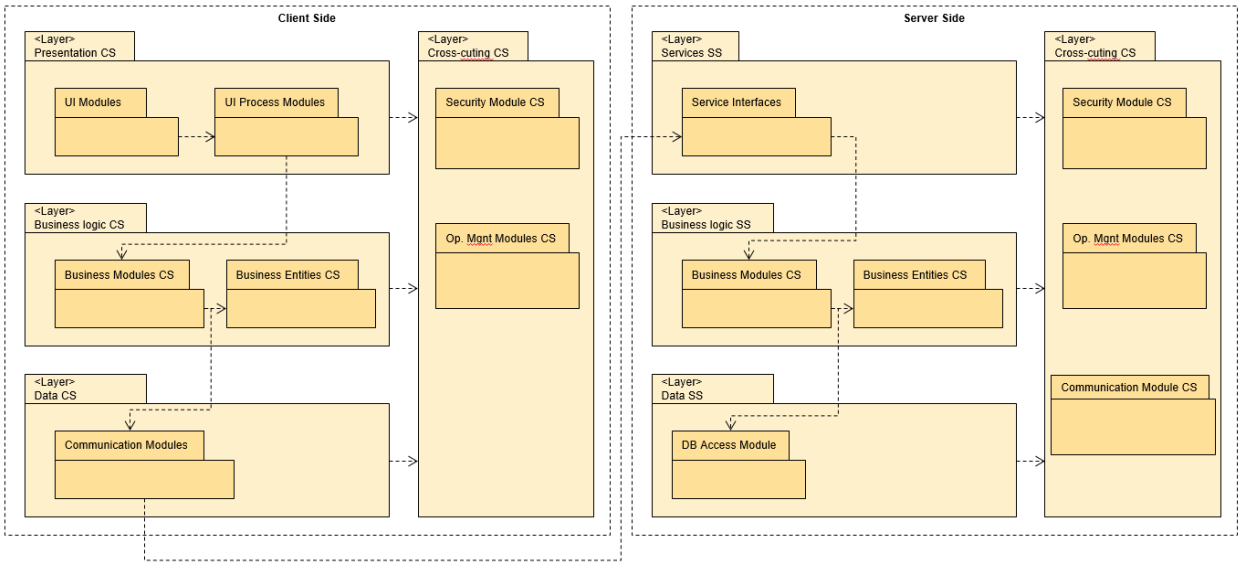

Figure 12. Module View

Element	Responsibility
Presentation Client side(CS)	이 계층에는 사용자 상호 작용 및 사용 사례 제어 흐름을 제어하는 모듈이 포함됩니다.
Business logic CS	이 계층에는 Client 측에서 내부적으로 실행할 수 있는 business logic operations 을 수행하는 모듈이 포함되어 있습니다.
Data CS	이 계층에는 서버와의 통신을 담당하는 모듈이 포함되어 있습니다.
Cross-cutting CS	이 계층에는 보안, 로깅 및 I / O 와 같은 여러 계층을 가로 지르는 기능이 있는 모듈이 포함됩니다. 이것은 드라이버 중 하나라도 CRN-6 을 달성하는 데 도움이 됩니다.
UI modules	이 모듈은 사용자 인터페이스를 렌더링하고 사용자 입력을 받습니다.
UI process modules	이 모듈은 모든 시스템 사용 사례 (화면 간 탐색 포함)의 제어 흐름을 담당합니다.
Business modules CS	이 모듈은 로컬에서 수행 할 수 있는 비즈니스 운영을 구현하거나 서버 측에서 비즈니스 기능을 노출합니다.
Business entities CS	이 엔티티는 도메인 모델을 구성합니다. 그들은 서버 측보다 덜 상세 할 수 있습니다.
Communication modules CS	이 모듈은 서버 측에서 실행되는 애플리케이션에서 제공하는 서비스를 사용합니다.
Services server side (SS)	이 계층에는 클라이언트가 사용하는 서비스를 노출하는 모듈이 포함되어 있습니다.
Business logic SS	이 계층에는 서버 측에서 처리해야하는 비즈니스 논리 작업을 수행하는 모듈이 포함되어 있습니다.
Data SS	이 계층에는 데이터 지속성 및 시간 서버와의 통신을 담당하는 모듈이 포함되어 있습니다. 이것은 QA-5 을 달성하는 데 도움이 됩니다.
Cross-cutting SS	이러한 모듈에는 보안, 로깅 및 I / O 와 같은 여러 계층에 걸친 기능이 있습니다.
Service Interfaces SS	이 모듈은 클라이언트가 사용하는 서비스를 노출합니다.
Business modules CS	이 모듈은 비즈니스 운영을 구현합니다.
Business entities CS	이 엔티티는 도메인 모델을 구성합니다.

DB access module	이 모듈은 관계형 데이터베이스에 대한 비즈니스 항목 (객체)의 지속성을 담당합니다. 그것은 객체 지향 관계형 매핑을 수행하고 지속성 세부 사항에서 응용 프로그램의 나머지 부분을 보호한다.
Time Server access module	이 모듈은 시간 서버와의 통신을 담당합니다. 다양한 유형의 시간 서버와의 통신을 지원하기 위해 시간 서버와의 작업을 격리하고 추상화합니다. (UC-4 참조).

III.Dynamic Behavior

III.1 Sequence Diagram

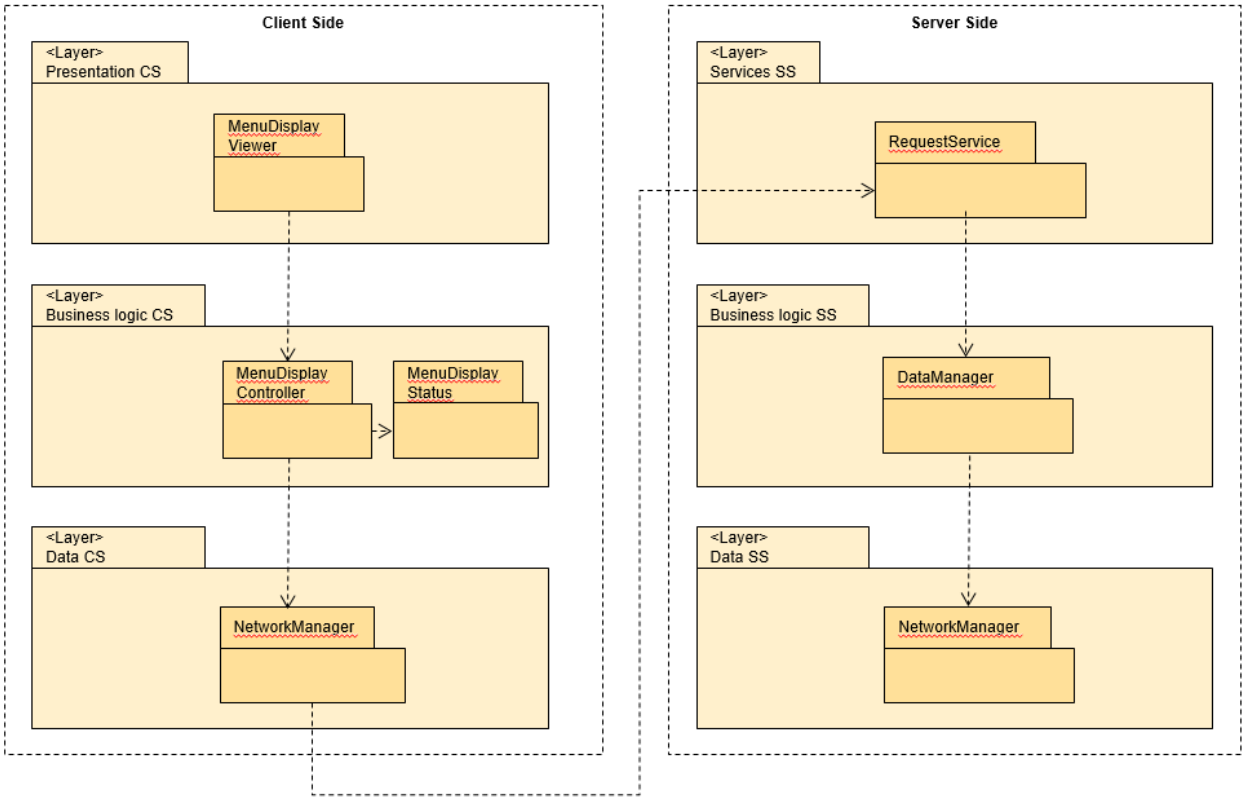

Figure 13. Module View (include specific elements)

III.1.1 UC-1: Manage database

총 음료의 개수는 20 종류이다.

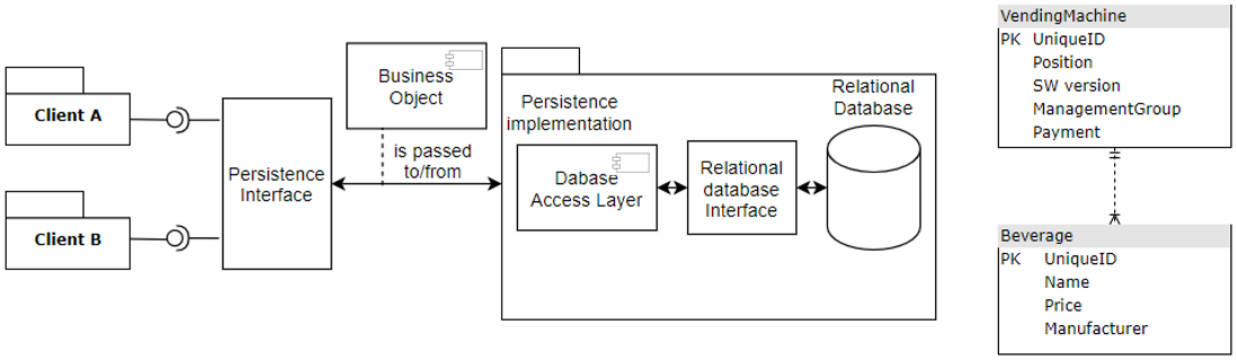

Figure 14.UC-1 Sequence Diagram -1

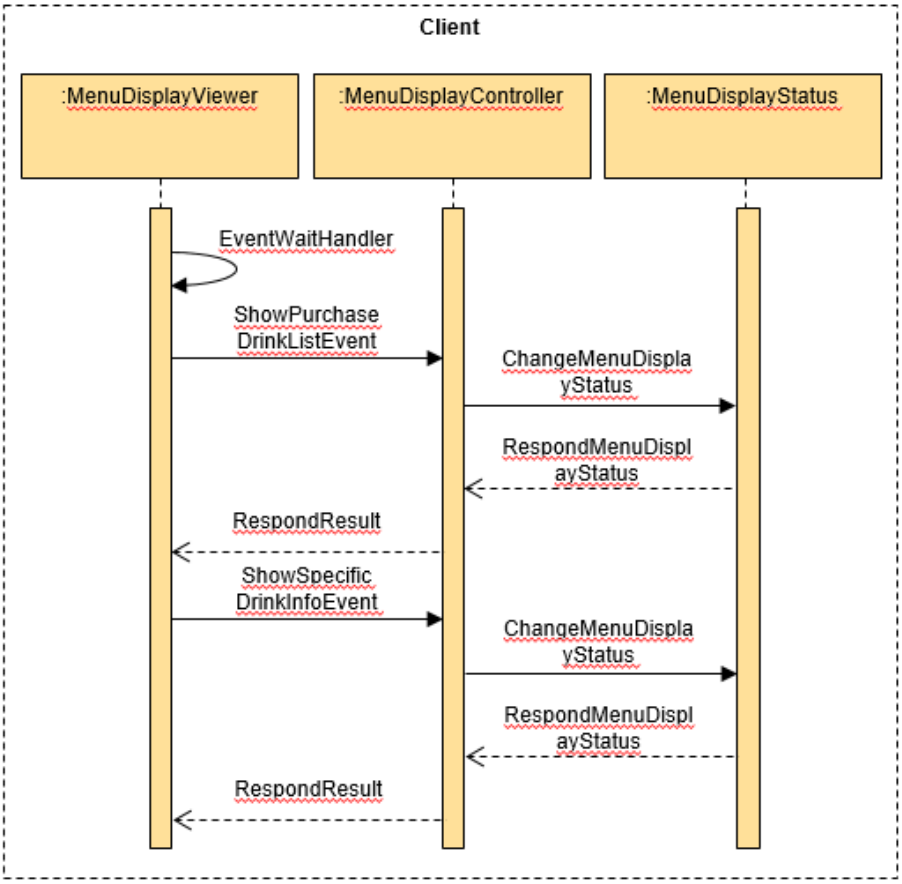

Figure 15. UC-1 Sequence Diagram-2

Method Name	Description
Element: MenuDisplayViewer	
EventWaitHandler	사용자가 이벤트를 발생시키는 것을 기다림
ShowPurchaseDrinkListEvent	구매할 수 있는 모든 음료리스트 보기 이벤트 발생
ShowSpecificDrinkInfoEvent	판매 음료 중 세부 정보 보기
Element: MenuDisplayController	
ChangeMenuDisplayStatus	내부 State machine 상태를 모든 음료 리스트 보기로 변경
RespondResult	사용자에게 모든 음료리스트 보기
Element: MenuDisplayStatus	
RespondMenuDisplayStatus	내부 State machine 상태 응답

III.1.2 UC-2: Display information

- 한 자판기는 7 종류의 음료 판매
- 판매하지 않는 음료도 메뉴는 제공

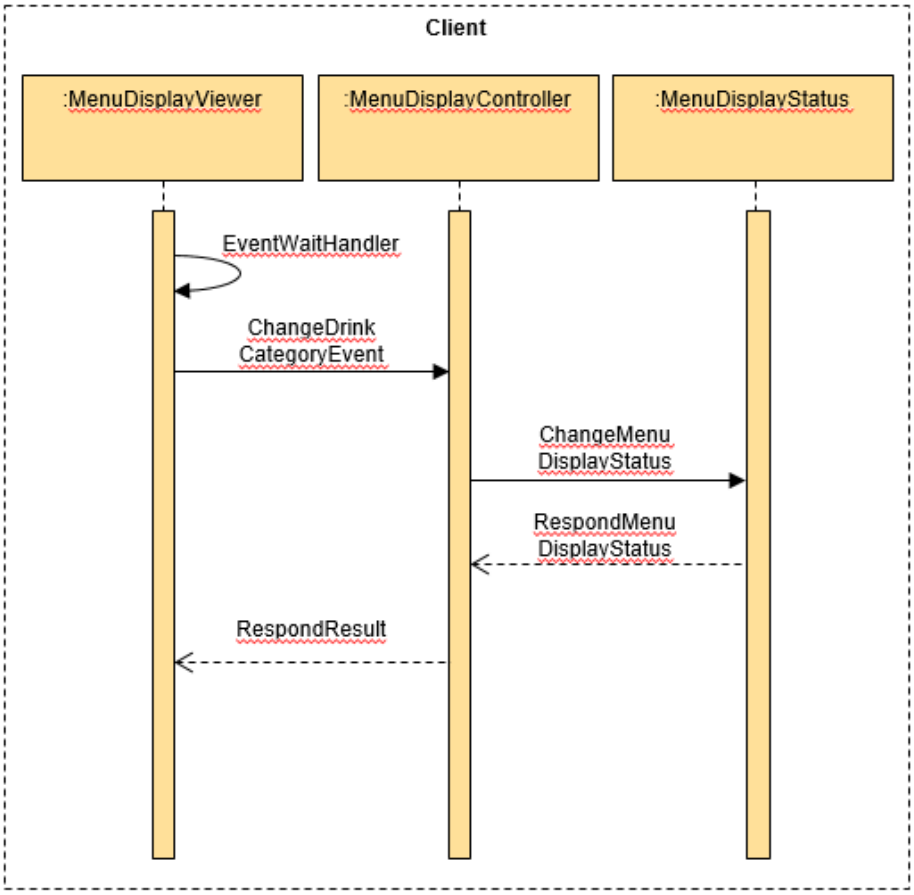

Figure 16. UC-2 Sequence Diagram

Method Name	Description
Element: MenuDisplayViewer	
EventWaitHandler	사용자가 이벤트를 발생시키는 것을 기다림
ChangeDrinkCategoryEvent	현재 자판기 판매하는(7 종류)/ 판매하지 않는, 이온/탄산 등 category 분류 이벤트 발생
Element: MenuDisplayController	
ChangeMenuDisplayStatus	Category 분류에 따른 내부 State machine 상태 변경
RespondResult	사용자에게 Category 에 따른 음료리스트 보기
Element: MenuDisplayStatus	
RespondMenuDisplayStatus	내부 State machine 상태 응답

III.1.3 UC-3: Process Tasks

- 사용자가 음료를 선택 후 결제하면 음료가 제공된다.
- 결제는 카드로 하고, 잔액이 부족한 경우 결제되지 않는다.

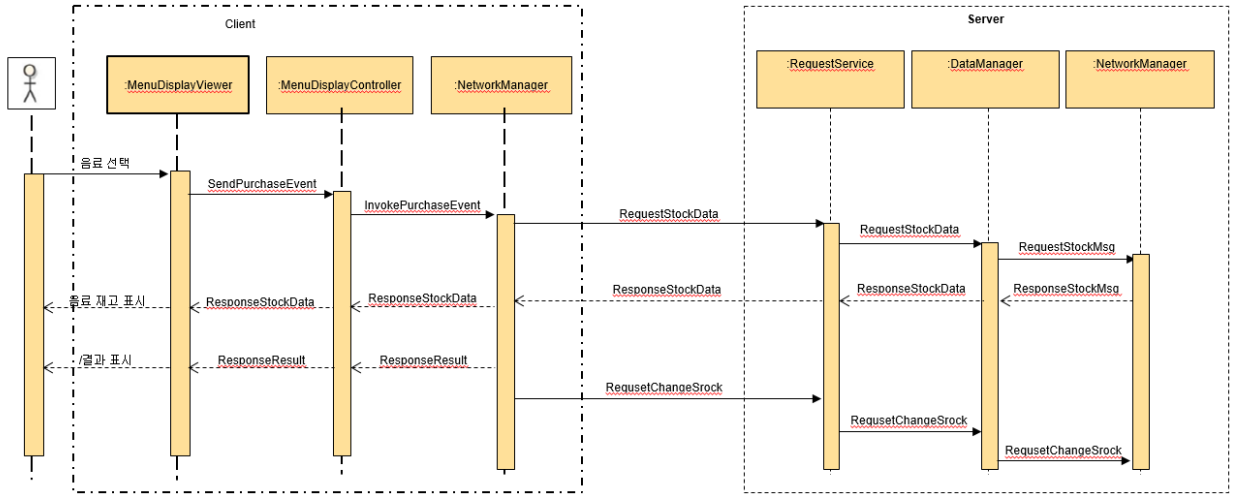

Figure 17. UC-3 Sequence Diagram

Method Name	Description
Element: MenuDisplayViewer	
SendPurchaseEvent	사용자 구매 요청 이벤트 발생
음료 재고 표시	음료 재고 표시
결과 표시	음료 구매에 대한 처리 결과 응답
Element: MenuDisplayController	
InvokePurchaseEvent	구매 요청 이벤트 호출
ResponseStockData	음료 재고에 대한 처리 결과 응답
ResponseResult	음료 구매에 대한 처리 결과 응답
Element: NetworkManager	
RequestStockData	서버에 음료 재고 정보 요청
ResponseStockData	서버에서 음료 재고 정보를 받아 응답
ResponseResult	음료 구매에 대한 처리 결과를 응답
RequsetChangeSrock	구매로 인한 재고 변동사항에 대한 DB 수정 요청
Element: RequestService	
RequestStockData	음료 재고 정보 요청
ResponseStockData	음료 재고 정보를 자판기에 응답
RequsetChangeSrock	구매로 인한 재고 변동사항을 DB 로 전달
Element: DataManager	
RequestStockMsg	DB 에 음료 재고 정보 요청
ResponseStockData	음료 재고 정보를 받아 응답
RequsetChangeSrock	구매로 인한 재고 변동사항에 대한 DB 수정 요청

Element: NetworkManager	
ResponseStockMsg	DB 에서 음료 재고 정보를 받아 응답

III.1.4 UC-4: Manage Network

- 음료 재고가 부족하거나 자판기에서 판매하지 않는 음료를 구매하려는 경우 다른 자판기 재고 확인 후 위치를 안내한다. 이 때, 네트워크 상의 자판기에 Broadcast MSG 를 통해 재고 확인을 요청하여 확인하고, 네트워크 MSG 를 통해 대상 자판기의 위치를 확인하여 안내한다.

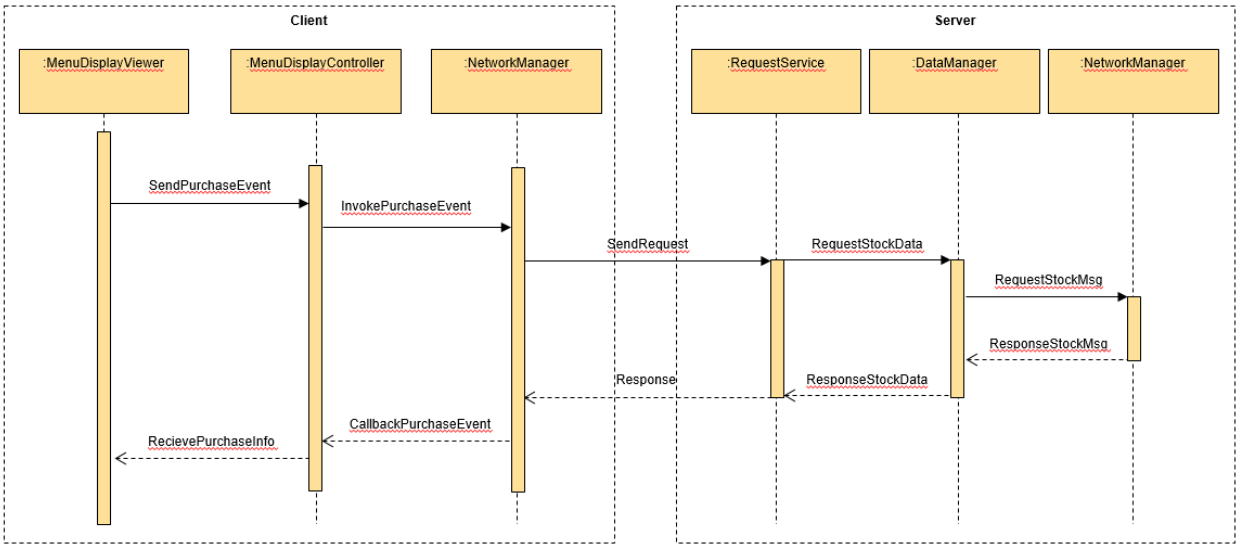

Figure 18. UC-4 Sequence Diagram

Method Name	Description
Element: MenuDisplayViewer	
SendPurchaseEvent	사용자 구매 요청 이벤트 발생
Element: MenuDisplayController	
InvokePurchaseEvent	구매 요청 이벤트 호출
RecievePurchaseInfo	사용자 구매 정보 받기
Element: NetworkManager	
SendRequest	구매한 음료 정보(수량, 비용 등) 요청
CallbackPurchaseEvent	구매 요청 이벤트 호출에 따른 콜백 함수 실행
Element: RequestService	
RequestStockData	구매 음료 재고 상태 송신
Response	요청에 따른 결과 응답
Element: DataManager	

RequestStockMsg	구매 음료 재고 상태 메세지 송신
ResponseStockData	구매 음료 재고 상태 수신
Element: NetworkManager	
ResponseStockMsg	구매 음료 재고 상태 메세지 수신

III.1.5 UC-5: Identification

- 다른 자판기의 음료 구매에 대해 선 결제를 할 수 있다. 이 때, 현재 자판기에서 결제 후 인증 코드를 발급하며 다른 자판기로 가서 인증코드를 입력하면 음료가 나온다.

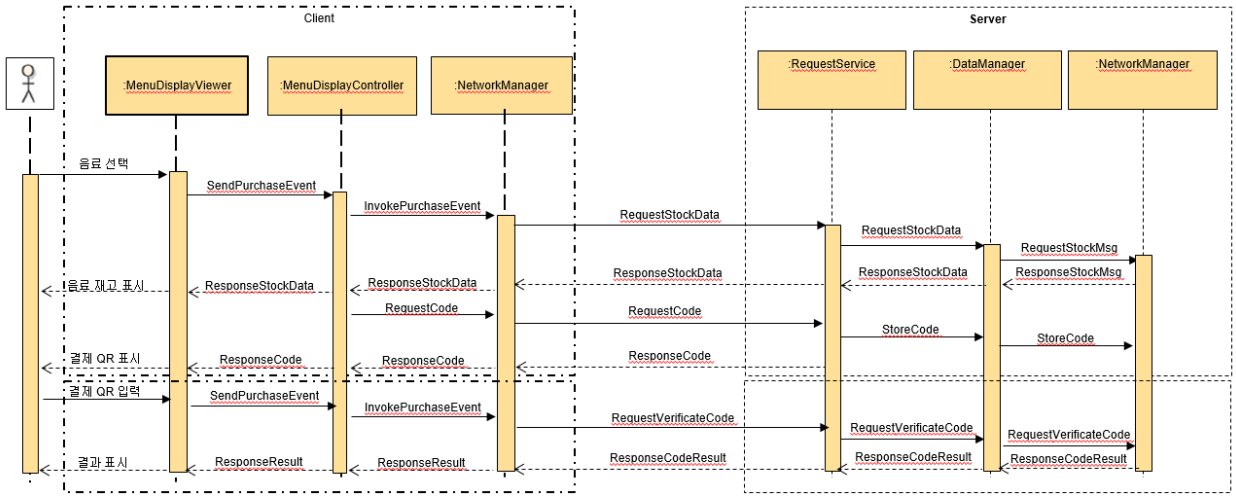

Figure 19. UC-5 Sequence Diagram

Method Name	**Description**
Element: MenuDisplayViewer	
SendPurchaseEvent	사용자 구매 요청 이벤트 발생
음료 재고 표시	음료 재고 표시
결재 QR 표시	결제 QR 을 사용자에게 표시
SendPurchaseEvent	사용자 구매 요청 이벤트 발생
결과 표시	음료 구매에 대한 처리 결과 응답
Element: MenuDisplayController	
InvokePurchaseEvent	구매 요청 이벤트 호출
ResponseStockData	음료 재고에 대한 처리 결과 응답
RequestCode	결제 코드 요청
ResponseCode	결제 코드 응답
InvokePurchaseEvent	구매 요청 이벤트 호출
ResponseResult	음료 구매에 대한 처리 결과 응답

Element: NetworkManager	
RequestStockData	서버에 음료 재고 정보 요청
ResponseStockData	서버에서 음료 재고 정보를 받아 응답
RequestCode	결제 코드 요청
ResponseCode	서버에서 결제 코드를 받아 전달
RequestVerificateCode	QR 코드에 대한 무결성 확인 및 구매에 따른 재고 변동 전달
ResponseResult	음료 구매에 대한 처리 결과 응답

Method Name	**Description**
Element: RequestService	
RequestStockData	음료 재고 정보 요청
ResponseStockData	음료 재고 정보를 자판기에 응답
StoreCode	결제 코드를 DB 에 저장
ResponseCode	결제 코드를 클라이언트에 전달
RequestVerificateCode	결제코드 무결성 확인 및 구매에 따른 재고 변동 전달
ResponseCodeResult	결제 코드 검사 결과 응답
Element: DataManager	
RequestStockMsg	DB 에 음료 재고 정보 요청
ResponseStockData	음료 재고 정보를 받아 응답
StoreCode	결제 코드를 DB 에 저장
RequestVerificateCode	결제코드 무결성 확인 및 구매에 따른 재고 변동 전달
ResponseCodeResult	결제 코드 검사 결과 응답
Element: NetworkManager	
ResponseStockMsg	DB 에서 음료 재고 정보를 받아 응답
ResponseCodeResult	결제 코드 검사 결과 응답

III.1.6 Perform analysis of current design and review iteration goal and achievement of design purpose

Not Addressed	**Partially Addressed**	**Completely Addressed**	**Design Decisions Made During the Iteration**
		UC-1	Reference architecture 를 sequence diagram 을 통해 고려 완료함
		UC-2	Reference architecture 를 sequence diagram 을 통해 고려 완료함
		UC-3	Reference architecture 를 sequence diagram 을 통해 고려 완료함

		UC-4	Reference architecture 를 sequence diagram 을 통해 고려 완료함
		UC-5	Reference architecture 를 sequence diagram 을 통해 고려 완료함
	QA-1		Reference architecture 와 Deploy pattern 을 선정함에 있어 부분적으로 고려되었음
	QA-2		Reference architecture 와 Deploy pattern 을 선정함에 있어 부분적으로 고려되었음
QA-3			
QA-4			
QA-5			
	QA-6		Server 의 Reference architecture 를 선정함에 있어 부분적으로 고려됨
		CRN-1	해당 Concern 을 고려하여 System 의 architecture 는 Green field 에서 설계되었음.
CRN-2			
	CRN-3		해당 Concern 을 고려하여 architecture 설계 과정에서 모든 Architectural Driver 를 고려함
		CRN-4	Sequence diagram 세분화 중 고려 완료
		CRN-5	Sequence diagram 세분화 중 고려 완료
		CRN-6	해당 Concern 을 반영하여 Server architecture 선정
		CON-1	부분 반영
		CON-2	Sequence diagram 을 통해 고려 완료함
		CON-3	Requirement 문서대로 반영

III.1.7 Refining Deployment diagram with new elements

- LoadBalancer
- TrapReciever

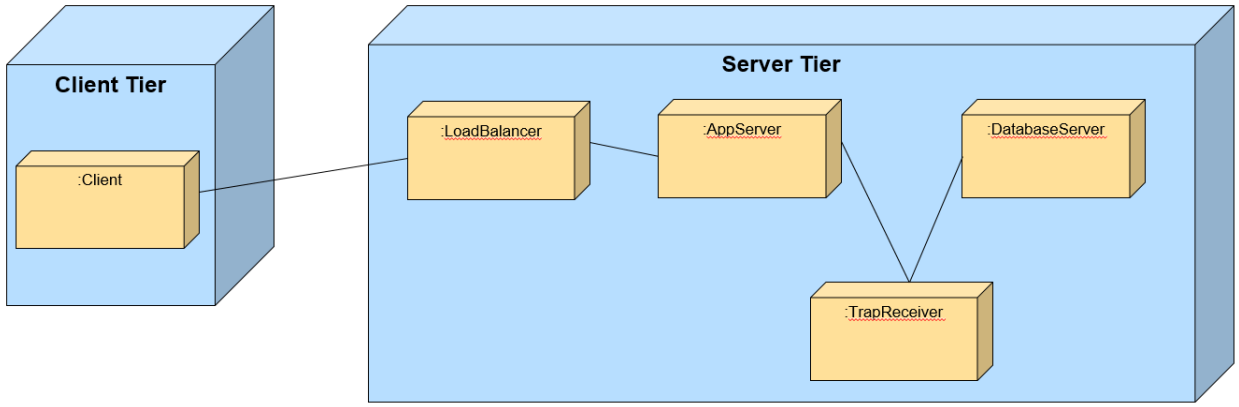

Figure 20. Refining Deployment Diagram

Illustrating how the TrapReceiver element exchanges messages with other elements to support UC-4 (Manage network)

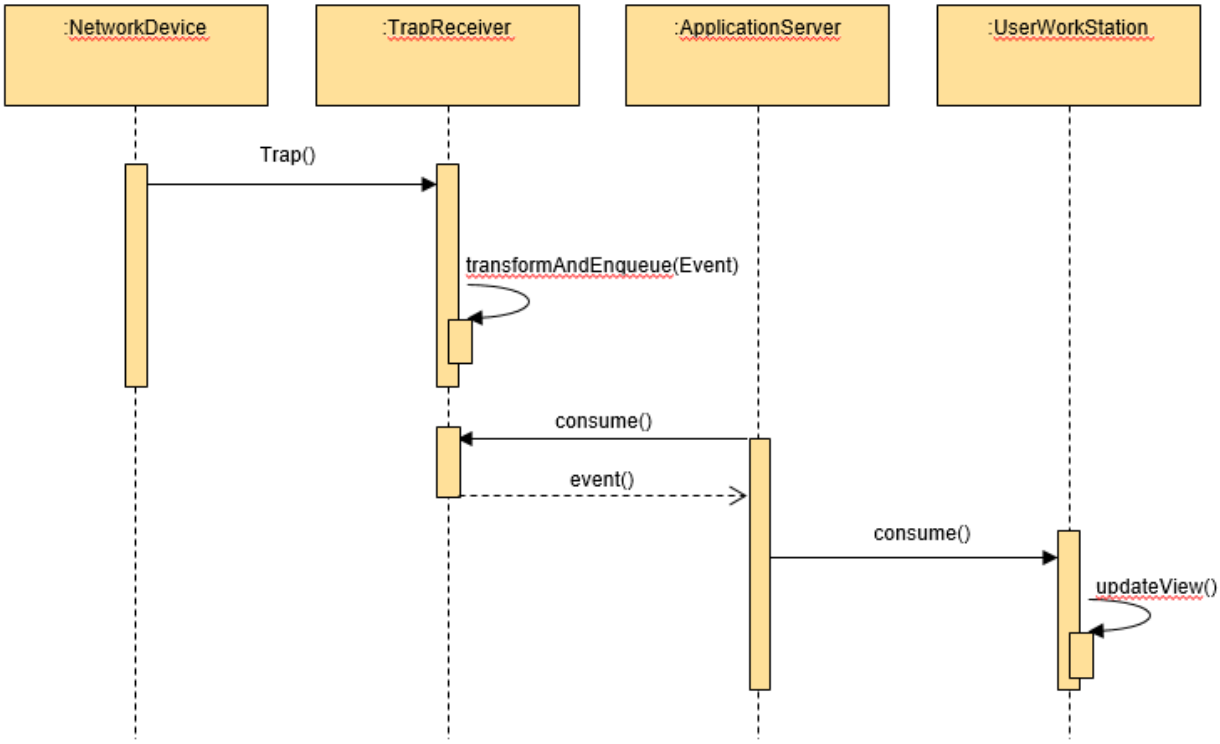

Figure 21. UC-4 Sequence Diagram(for refinement)

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
		UC-1	
		UC-2	
		UC-3	
		UC-4	
		UC-5	
	QA-1		
	QA-2		
	QA-3		Tactic 채택을 통해 부분 고려되었음
	QA-4		Tactic 채택을 통해 부분 고려되었음
		QA-5	Important QA 로 선택되어 Tactic, queue, framework 를 통해 고려 완료
	QA-6		
		CRN-1	
		CRN-2	ADD iteration 을 끝냄으로써 고려 완료
	CRN-3		
		CRN-4	
		CRN-5	
		CRN-6	
		CON-1	
		CON-2	
		CON-3	

IV.Conceptual Framework

IV.1 User Interface Framework

Portable UI 개발에 적합한 React Native Framework 를 선정하였다.

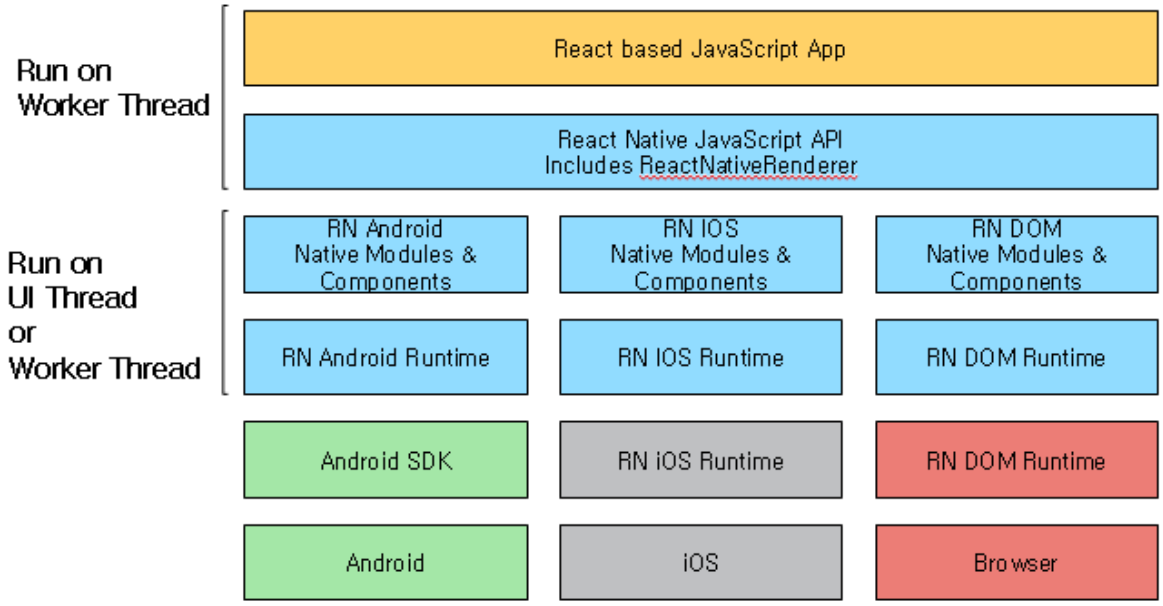

Figure 22. React Native Framework

Division	React Native Description
Technology family	Local user interface
Language	JavaScript
URL	https://reactnative.dev/
Purpose	Framework to support the creation of portable local user interface.
Overview	React Native is an open source application framework developed by Facebook. With react native, you can develop real native apps. By using JavaScript and React library, you can develop for both Android and iOS. ** Characteristic 1. A collection of "special" React components 2. Components compiled to Native Widgets 3. Native platform APIs exposed to JavaScript
Development Environment	React Native CLI
Benefits	1. Productivity: When the source code is modified, the changed contents can be checked immediately. 2. Open Source: MIT License
Limitations	1. Performance: A hybrid app method that uses a native bridge to connect a JavaScript thread and a native thread. lower than the native development method. 2. Native Functions Development: Services with many unique native features can be a bit difficult to develop.

V. Conclusion

DVM 시스템 개발을 하기 위해 일반적인 Architecture 이해와 구조를 분석해 보고, DVM 의 전반적인 시스템 구성 및 시나리오에 맞는 Architecture Structure 를 구성해보았다. 개발에 필요한 세부적인 개발환경 구축 및 Reference 자료를 추가가 필요하지만 이 문서를 통해 DVM 에 대한 시스템의 간소화, 전체적인 시스템, 시나리오를 통한 데이터 & 네트워크 & SW 흐름을 파악할 수 있었다. 1 장에서는 DVM 시스템의 기본적인 설명뿐만 아니라 Stakeholder 에 대한 역할, 요구사항, QA(Quality Attribute)에 대한 내용도 포함되어 있다. 2 장에서는 전반적인 DVM 시스템에 적합한 Reference Architecture 및 Deploy Pattern 등을 선정 및 그에 대한 설명에 포함되어 있다. 3 장에서는 실질적인 SW 시스템 시나리오에 대한 내용이 포함되어 있다. 4 장에는 개발에 필요한 SW Framework 가 포함되어 있다. 이 문서를 통해 Stakeholder 들에게 일관적인 시스템 이해를 주어지게 될 것이다.