Title:
Architecture Description by SEI

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. Architecture Description by SEI

I.1 Module View Type – Decomposition, Uses and Layered

I.1.1 Primary Presentation

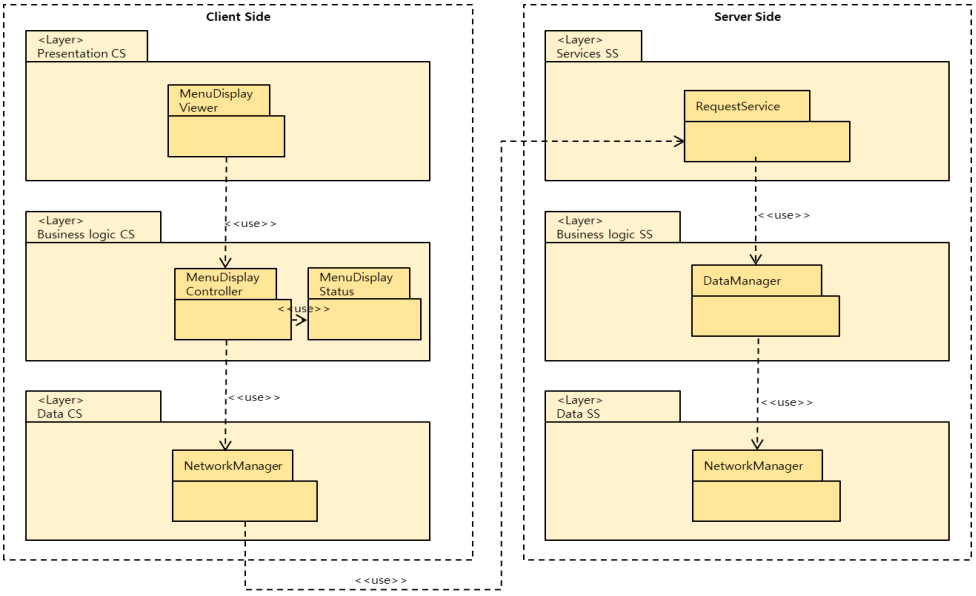


Figure 1. System Module View

I.1.2 Element Catalog

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).

I.1.3 Context Diagram

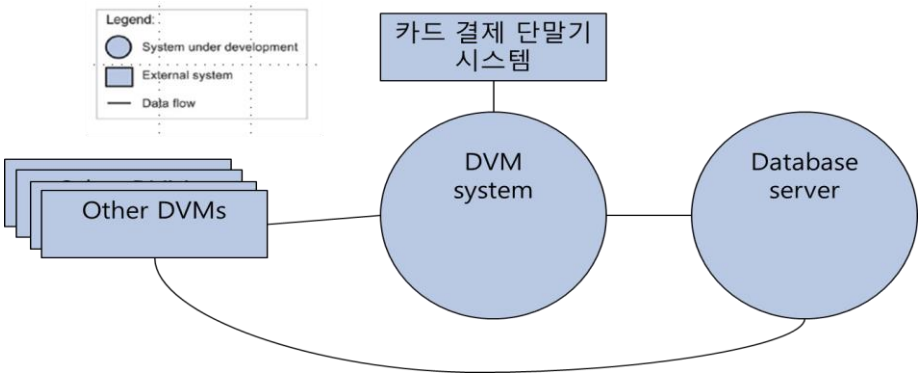


Figure 2. Overall System Diagram

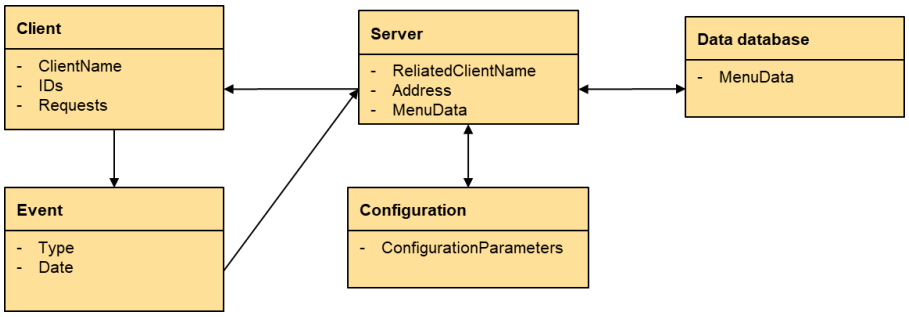


Figure 3. Initial Domain Model

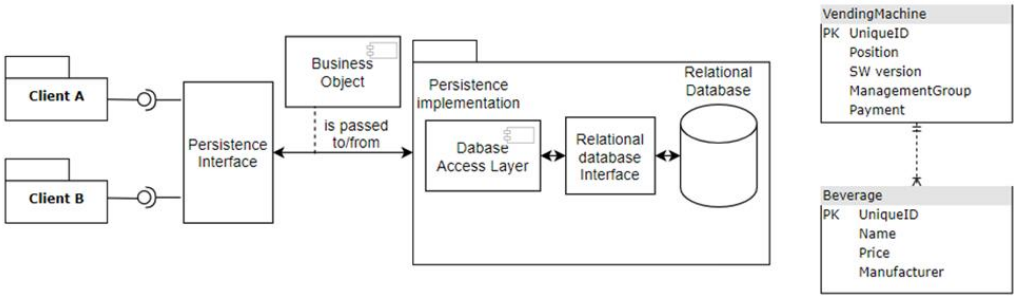


Figure 4. Database Diagram

I.1.4 Variability Guide

None

I.1.5 Rationale

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
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
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.
Create a **Domain model**	기능 세분화 전에 시스템에 대한 초기 domain model 을 생성하여 도메인의 주요 elements 를 식별해야 한다.
Identify **Domain Objects** that map to functional requirements	Use cases 를 빌딩 블록 하나에 캡슐화하기 위해 domain objects 를 결정
Decompose Domain Objects into general and specialized Components	생성한 Domain objects 를 각각 계층 내에 있는 세분화된 elements 로 분해가 필요하다.

I.2 Component and Connector View – Pipe and Filter Style

I.2.1 Primary Presentation

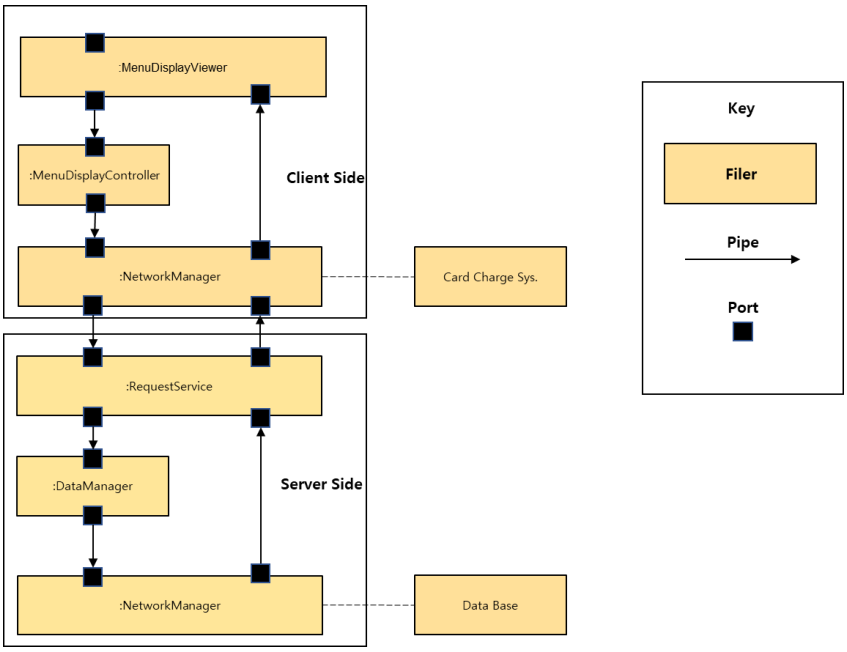


Figure 5. System Pipe and Filter View

I.2.2 Element Catalog

Element	Responsibility
MenuDisplayViewer	이 필터에는 사용자 상호 작용 및 사용 사례 제어 흐름을 제어하는 모듈이 포함됩니다.
MenuDisplayController	이 필터에는 Client 측에서 내부적으로 실행할 수 있는 business logic operations 을 수행하는 모듈이 포함되어 있습니다.
NetworkManager	이 필터에는 서버와의 통신을 담당하는 모듈이 포함되어 있습니다.
RequestService	이 필터에는 클라이언트가 사용하는 서비스를 노출하는 모듈이 포함되어 있습니다.
DataManager	이 필터에는 서버 측에서 처리해야하는 비즈니스 논리 작업을 수행하는 모듈이 포함되어 있습니다.
NetworkManager	이 필터에는 데이터 지속성 및 시간 서버와의 통신을 담당하는 모듈이 포함되어 있습니다. 이것은 QA-5 을 달성하는 데 도움이 됩니다.
Card Charge Sys.	카드 결제를 위한 VAN 통신을 담당하는 모듈이 포함되어 있습니다.
Data Base	서버의 데이터베이스입니다.

I.2.3 Context Diagram

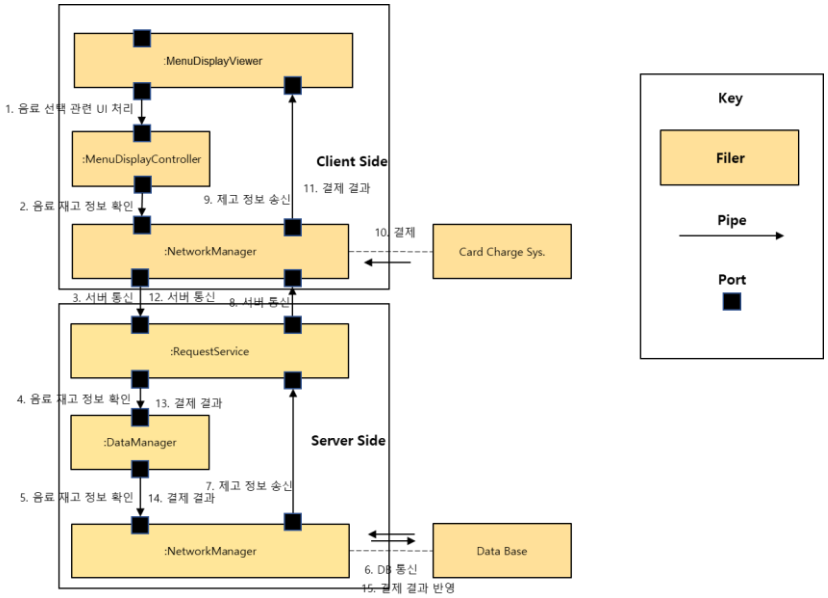


Figure 6. Element Behavior

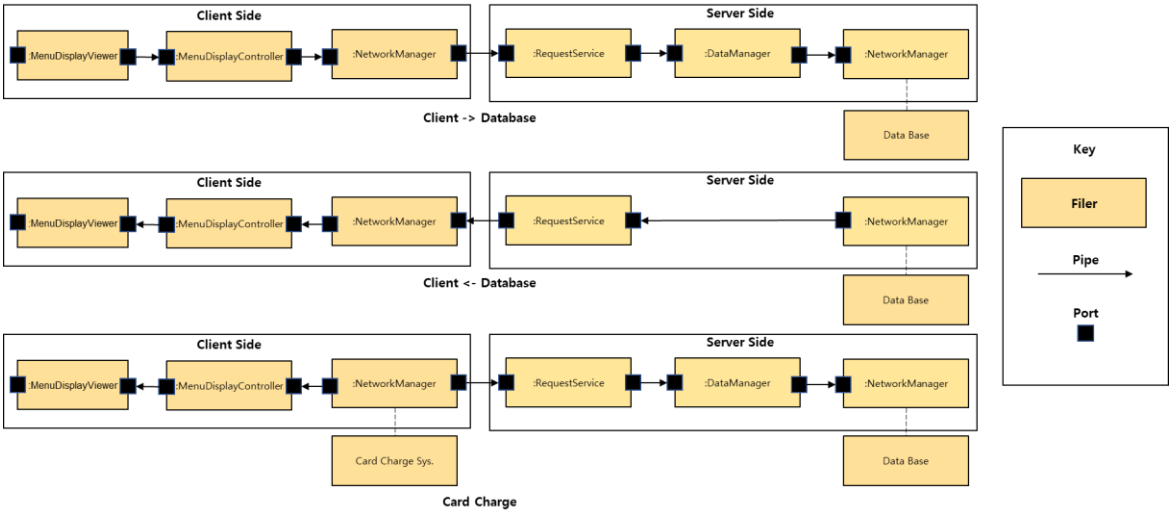


Figure 7. Data Flow

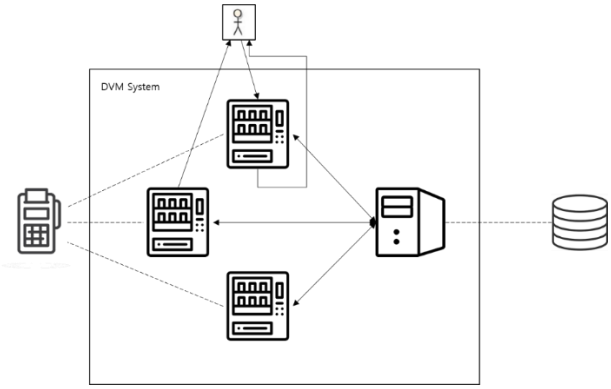


Figure 8. System Configuration

I.2.4 Variability Guide

None

I.2.5 Rationale

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
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
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.
Create a **Domain model**	기능 세분화 전에 시스템에 대한 초기 domain model 을 생성하여 도메인의 주요 elements 를 식별해야 한다.
Identify **Domain Objects** that map to functional requirements	Use cases 를 빌딩 블록 하나에 캡슐화하기 위해 domain objects 를 결정
Decompose Domain Objects into general and specialized Components	생성한 Domain objects 를 각각 계층 내에 있는 세분화된 elements 로 분해가 필요하다.

I.3 Component and Connector View – Other View

I.3.1 Primary Presentation

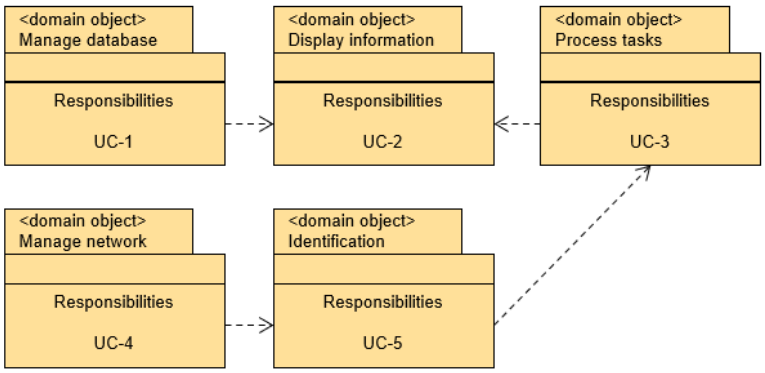


Figure 9 Domain objects associated with use cases

I.3.2 Element Catalog

I.3.2.1 UC-1: Manage database

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

I.3.2.2 UC-2: Display information

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

I.3.2.3 UC-3: Process tasks

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 변동사항애 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 변동사항애 대한 DB 수정 요청
Element: NetworkManager	
ResponseStockMsg	DB 에서 음료 재고 정보를 받아 응답

I.3.2.4 UC-4

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

I.3.2.5 **UC-5**

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

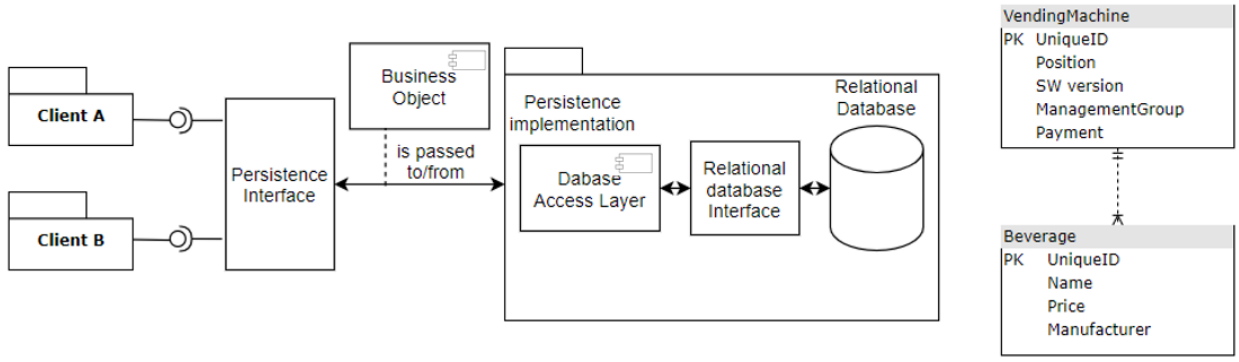
ResponseCode	서버에서 결제 코드를 받아 전달
RequestVerificateCode	QR 코드에 대한 무결성 확인 및 구매에 따른 재고 변동 전달
ResponseResult	음료 구매에 대한 처리 결과 응답

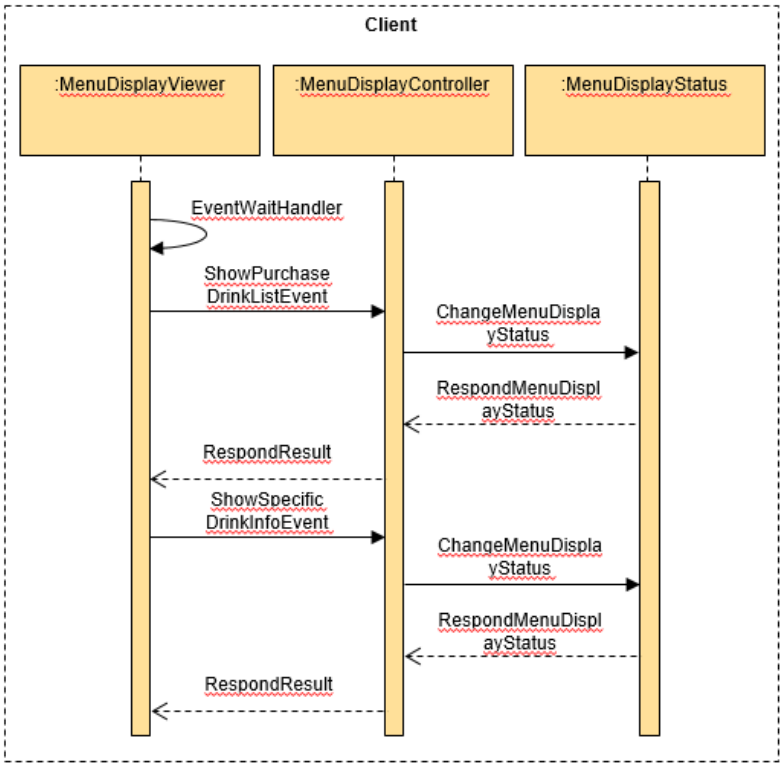
Method Name	**Description**
Element: RequestService	
RequestStockData	음료 재고 정보 요청
ResponseStockData	음료 재고 정보를 자판기에 응답
StoreCode	결제 코드를 DB 에 저장
ResponseCode	결제 코드를 클라이언트에 전달
RequestVerificateCode	결제코드 무결성 확인 및 구매에 따른 재고 변동 전달
ResponseCodeResult	결제 코드 검사 결과 응답
Element: DataManager	
RequestStockMsg	DB 에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

I.3.3 Context Diagram

I.3.3.1 **UC-1: Manage database**

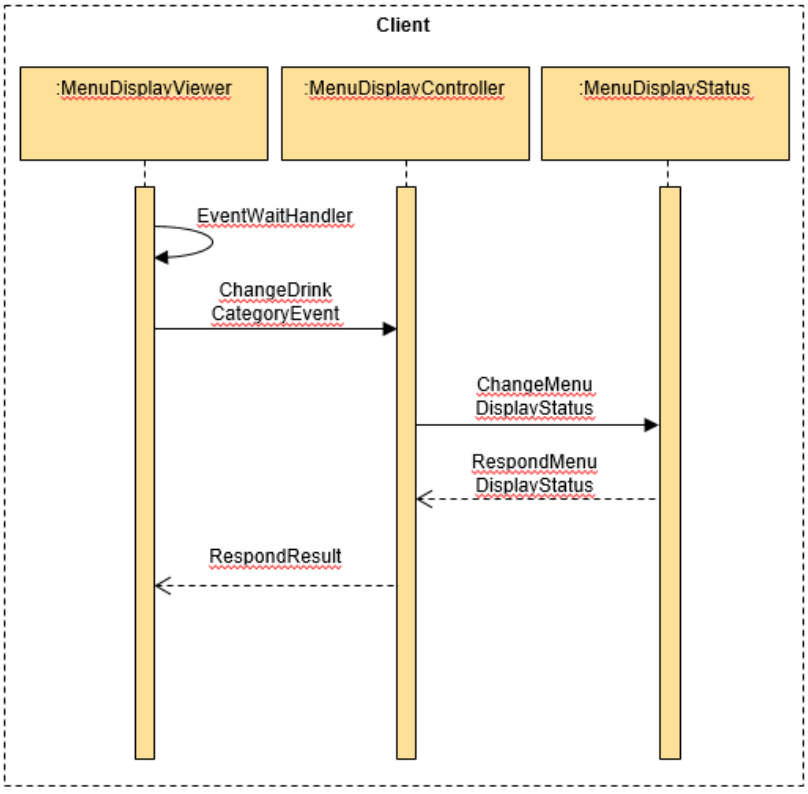
총 음료의 개수는 20 종류이다.





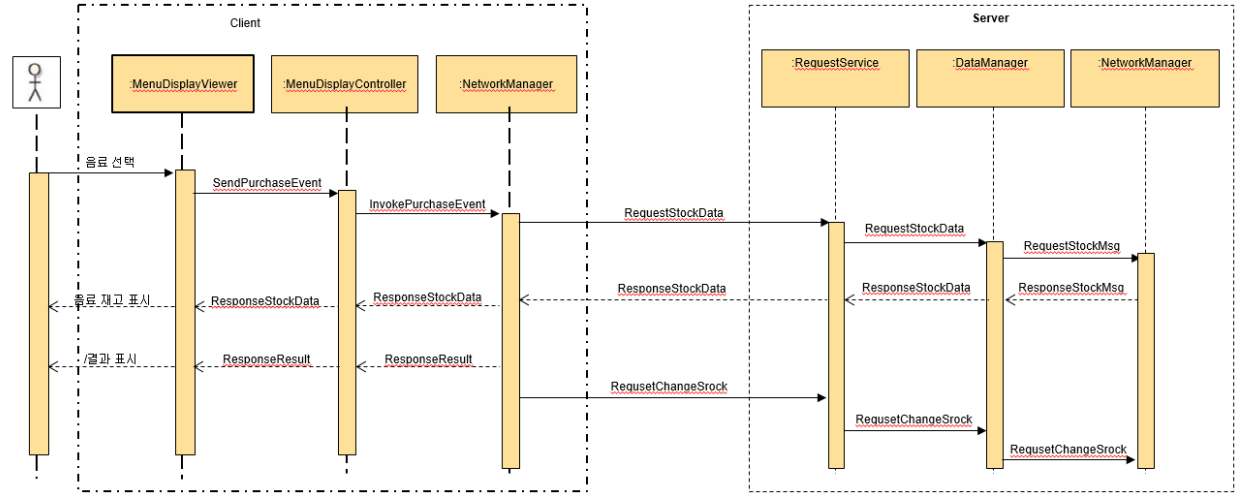
I.3.3.2 **UC-2: Display information**

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



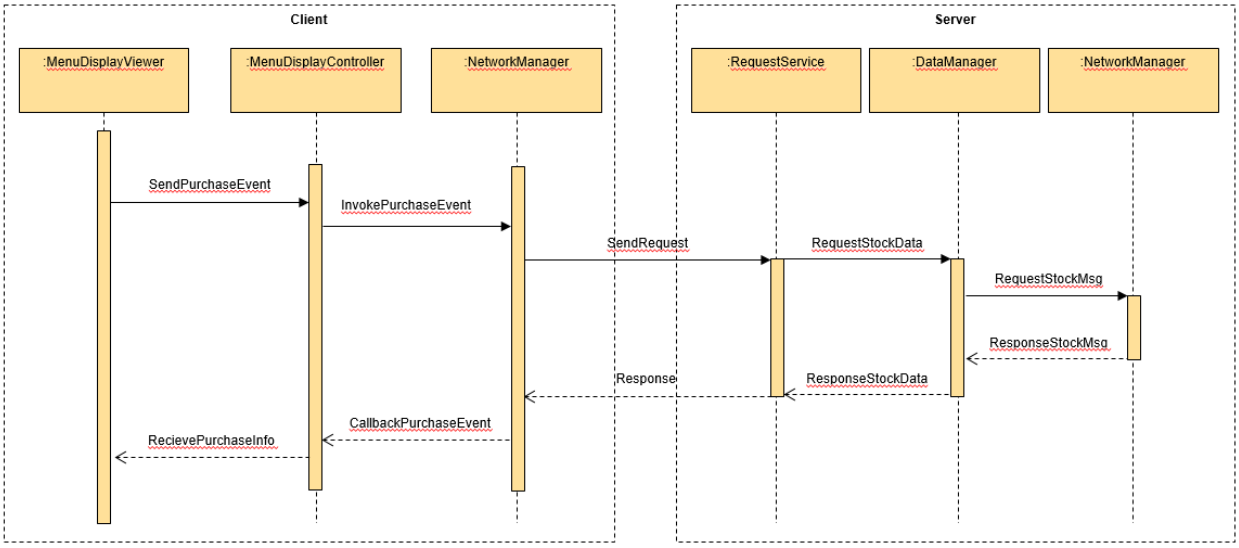
I.3.3.3 **UC-3: Process tasks**

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



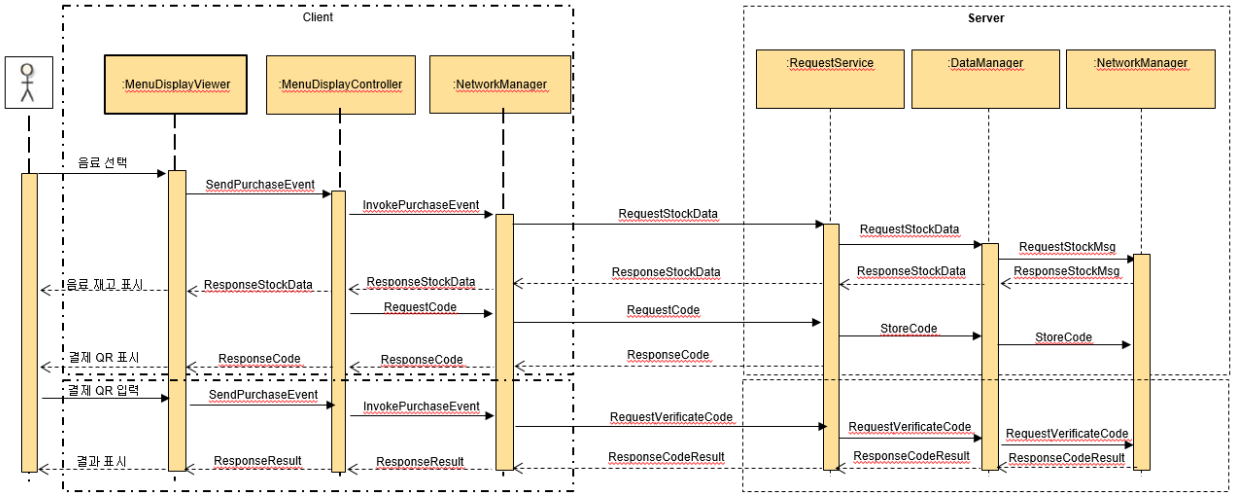
I.3.3.4 **UC-4: Manage Network**

- 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.



I.3.3.5 **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.



I.3.4 Variability Guide

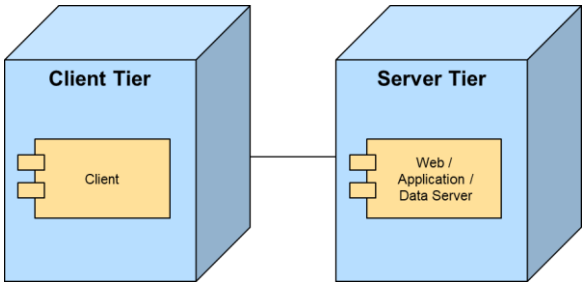
None

I.3.5 Rationale

Design Decisions and Location	**Rationale**
UC-1: Manage database	UC-1 을 수행하기 위해 software components 를 MenuDisplayViewer, MenuDisplayController, MenuDisplayStatus 로 구성
UC-2: Display information	UC-2 을 수행하기 위해 software components 를 MenuDisplayViewer, MenuDisplayController, MenuDisplayStatus 로 구성
UC-3: Process tasks	UC-3 을 수행하기 위해 software components 를 MenuDisplayViewer, MenuDisplayController, NetworkManager (client), RequestService, DataManager, NetworkManager (server)로 구성
UC-4: Manage network	UC-4 을 수행하기 위해 software components 를 MenuDisplayViewer, MenuDisplayController, NetworkManager (client), RequestService, DataManager, NetworkManager (server)로 구성
UC-5: Identification	UC-5 을 수행하기 위해 software components 를 MenuDisplayViewer, MenuDisplayController, NetworkManager (client), RequestService, DataManager, NetworkManager (server)로 구성

I.4 Allocation View Type - Deployment

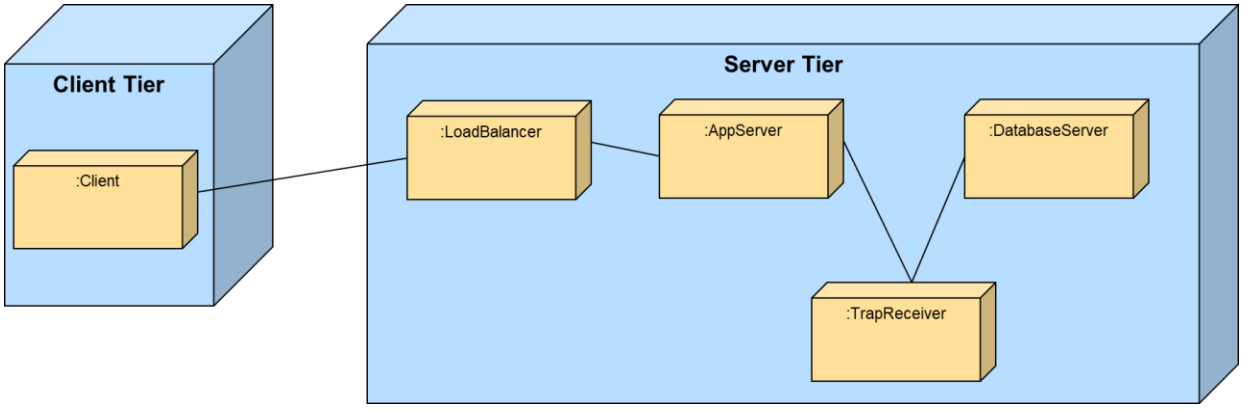
I.4.1 Primary Presentation



I.4.2 Element Catalog

Element	Responsibility
Client Tier	클라이언트 및 클라이언트 HW. 클라이언트는 인터넷에 연결되어 있어 이를 통해 서버와 통신.
Server Tier	서버 및 서버 HW. 클라이언트의 요청에 대해 응답 및 데이터베이스와 통신.

I.4.3 Context Diagram



I.4.4 Variability Guide

None

I.4.5 Rationale

Design Decision and Location	Rationale
Physically structure the application using the Two-tier Deployment pattern	**QA-1, 2 을 고려하여 배포할 시스템의 아키텍쳐는 간단할 수록 좋고, 각각의 DVM 은 Web application 을 거치지 않고 직접 database server 에 접근하므로 two-tier deployment pattern 이 적절하다고 판단**
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.**

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**

I.5 Documentation Beyond View

I.5.1 Documentation Roadmap

TABLE OF CONTENTS

I.5.2 How a View is Documented

I.5.2.1 Module View Type – Decomposition, Uses and Layered

Module View 는 Decomposition, Uses and Layered Type View 를 통합하여 작성되었다. 시스템의 정적인 구성을 하나로 통합하여 나타내었으며 앱 개발자, 네트워크 개발자, 유지보수 관리자를 고려하여 작성되었다.

I.5.2.2 Component and Connector View – Pipe and Filter Style

Component and Connector View 는 Pipe and Filter Style 과 Sequence Style 로 작성되었다. 그 중 Pipe and Filter Style 은 시스템의 동적인 부분, 그 중 데이터의 흐름을 중점으로 나타내었으며, 마케팅 담당자, 네트워크 개발자, 유지보수 관리자를 고려하여 작성되었다.

I.5.2.3 Component and Connector View – Sequence Style

Component and Connector View 는 Pipe and Filter Style 과 Sequence Style 로 작성되었다. 그 중 Sequence Style 은 각 Use Case 에 따른 시스템의 동적인 부분을 나타내었으며, 마케팅 담당자, 네트워크 개발자, 유지보수 관리자를 고려하여 작성되었다.

I.5.2.4 Allocation View Type - Deployment

Allocation View 는 현 시점에서 가장 명확하게 작성될 수 있고, 소요가 높다고 판단된 Deployment View 로 작성되었다. 이는 사장, 마케팅 담당자, 네트워크 개발자를 고려하여 작성되었다.

I.5.3 System Overview

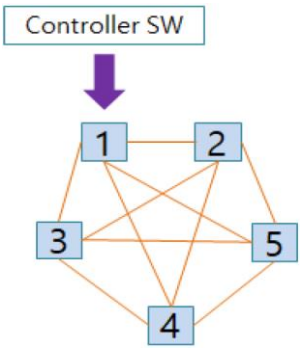


Figure 10. 분산 자판기 시스템 다이어그램

분산 자판기 시스템의 전체적인 구성을 보면 위와 같이 개략적으로 나타낼 수 있으며, 본 시스템은 위 다이어그램에서 Controller SW 에 해당한다. 해당 시스템에 요구된 기능적 요소와 Architectural Driver 는 다음과 같다.

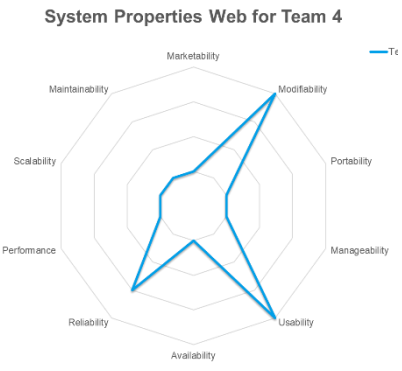
I.5.3.1 Primary Functional Requirement

Design Decisions and Location	Rationale
UC-1: Manage database	총 음료의 개수는 20 종류이다.
UC-2: Display information	한 자판기는 7 종류의 음료를 판매하며, 판매하지 않는 음료도 메뉴는 제공한다.
UC-3: Process tasks	사용자가 음료를 선택 후 결제하면 음료가 제공된다. 결제는 카드로 하고, 잔액이 부족한 경우 결제되지 않는다.
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 msg를 통해 재고 확인을 요청하여 확인하고, 네트워크 msg를 통해 대상 자판기의 위치를 확인하여 안내한다.
UC-5: Identification	다른 자판기의 음료 구매에 대해 선결제를 할 수 있다. 이 때, 현재 자판기에서 결제 후 인증 코드를 발급하며 다른 자판기로 가서 인증코드를 입력하면 음료가 나온다.

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

I.5.3.2 Major Architectural Objectives and Functions

Stakeholders (역할 담당)	주요 역할	희망사항-Goal	관련 QA
사장님 (배재욱)	사업 총괄 / 해당 사업의 시장성, 기술적 측면, 사업전개능력, 예상 경쟁력, 경제성 등을 판단하여 **사업의 진행방향을 결정**한다.	- 산출물이 시장성을 가져야 하고, 개발 비용이 최소화되어야 한다. - 시장의 수요가 변화할 때 개발 제품의 수정이 용이해야 한다.	(1) Marketability (2) Modifiability
마케팅 담당자 (배재욱)	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. - SW의 기능이 분명하고 간결하며 사용이 편리해야 한다.	(1) Portability (2) Manageability (3) Usability
사용자 (이태영)	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 안내해줘야 한다. - 인증 코드는 범용성 있는 것을 사용했으면 한다.	(1) Usability (2) Usability (3) Usability
앱 개발자 (이태영)	사용자에게 서비스를 제공하기 위해 시스템 개발한다.	- 자판기 작동 시 일주일 정도는 이상 없이 동작해야 한다. - 예외처리 케이스 발생 시에도 자판기가 동작될 수 있어야 한다. - 화면 사이의 연결이나 연동이 사용자로 하여금 불편함을 느끼지 않도록 0.5초 이내로 제한한다. - 사용자가 쓴 금액이 손실되지 않도록 시스템이 동작해야 한다.	(1) Availability (2) Reliability (3) Performance (4) Reliability
네트워크 개발자 (김상권)	네트워크 시스템을 주어진 요구사항에 따라 설계하고 구현한다.	- 네트워크가 끊긴 경우에도 로컬에서 동작 가능했으면 좋겠다. - 네트워크 프로토콜은 간단하여 수정하기 쉬웠으면 한다. - 여러 이용자가 동시에 사용해도 네트워크는 크게 영향 받지 않았으면 한다.	(1) Reliability (2) Modifiability (3) Scalability
유지보수 담당자 (김상권)	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) Maintainability



I.5.3.3 Quality Attribute and Architectural Concern

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 Driver를 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 DBMS로 데이터베이스를 구현
CRN-6	웹 방화벽 구축, 침입탐지시스템 등 정보보호를 위한 관리적 기술적 물리적인 시스템 구축

Quality Attribute	Scenario	Associated Use Case
QA-1: Marketability	기존 코드 중 재활용 가능한 부분의 활용을 통헤 낮은 개발 비용/ 빠른 개발속도를 달성해야 한다.	ALL
QA-2: Usability	SW의 기능이 분명하고 간결하며 사용이 편리해야 한다.	UC-3, 4, 5
QA-3: Usability	모든 구매한 물품과 비용이 한 눈에 보일 수 있게 해야 한다.	UC-1, 2
QA-4: Performance	화면 사이의 연결이나 연동이 사용자로 하여금 불편함을 느끼지 않도록 0.5초 이내로 제한한다.	UC-3, 4, 5
QA-5: Availability	자판기 작동 시 일주일 정도는 이상 없이 동작해야 한다.	ALL
QA-6: Modifiability	음료 메뉴 추가나 삭제가 용이했으면 좋겠다.	UC-1

I.5.4 Mapping Between Views

I.5.4.1 Mapping from C&C View to Module View

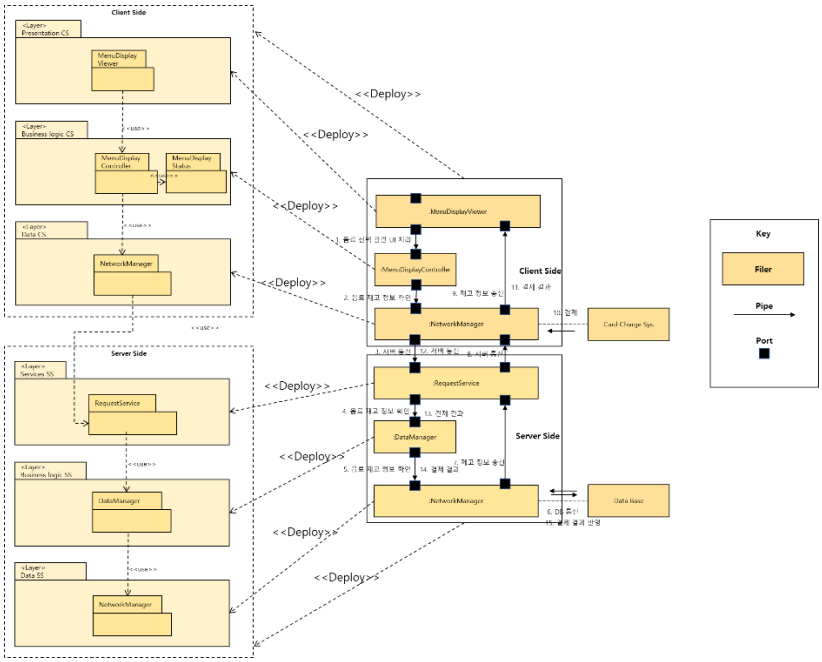


Figure 11. Mapping from C&C View to Module View

I.5.4.2 Mapping from Decomposition View to Layered View

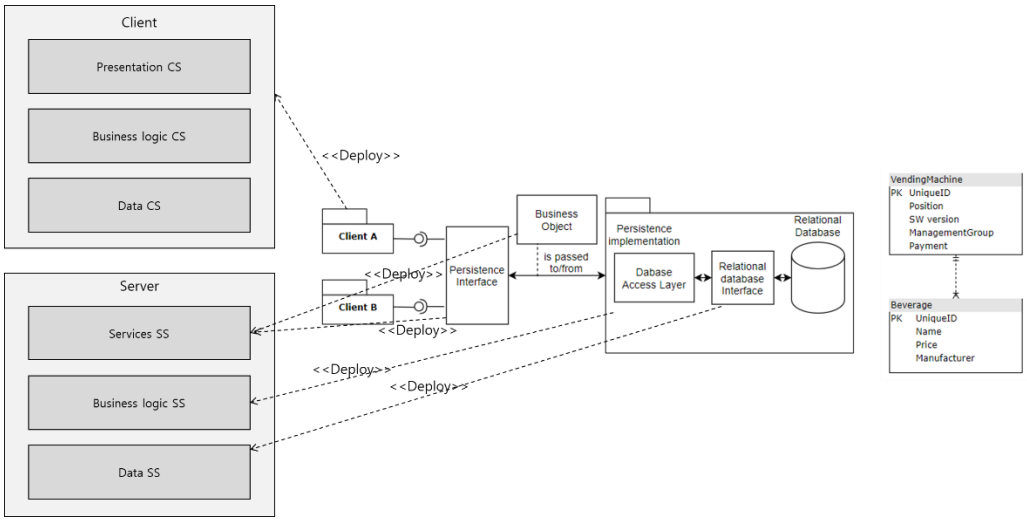


Figure 12. Mapping from Decomposition View to Layered View

I.5.5 Rationale

	Module Views				C&C View	Allocation Views			
	Decomposition	Uses	Generalization	Layered	Various	Deployment	Implementation	Install	Work Assignment
사장	s	s		s		d			d
마케팅 담당자	s				d	d		d	
사용자					s	o			
앱 개발자	d	d	d	d	d				
네트워크 개발자	s	d	s	d	d	d			
유지보수 담당자	d	d	d	d	s	s	s	s	
Key : d = detail, s = some, o = overview,									

위 표에 정리된 것과 같이 Skakeholder 의 각 필요성에 따라 여러 View Type 을 사용하여 문서를 구성하였다. Module Views 에선 가장 소요가 높은 Decomposition, Uses, Layered View Type 을 하나로 통합하여 작성하였으며 Allocation View 에서는 현 시점에서 명확히 작성 가능하며 가장 소요가 높은 Deployment View Type 을 작성하였다. C&C View 의 경우 필요성에 따라 Pipe and Filter Style 과 Other View Style 을 적용하였다.