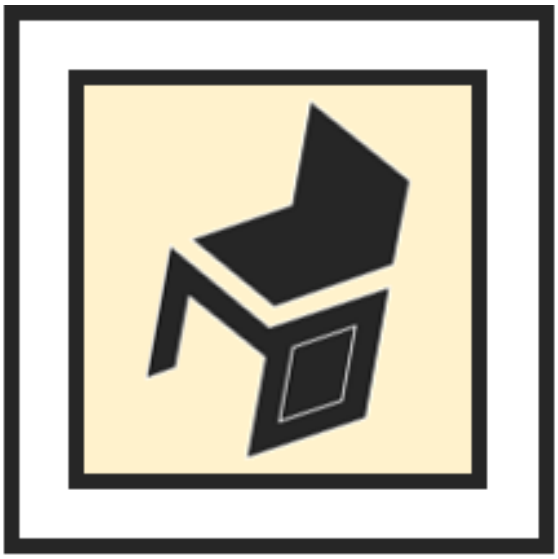
Team 6 - InteReal

- Design Specification -



INTEREAL

Student Number	Name
2013314453	유정엽
2013310718	서해빈
2015312076	양승호
2017310457	김현주

Contents

1 Preface

이 장에서는 문서의 예상 독자를 정의하고 각 장의 내용을 간략하게 소개한다. 또한 문서의 Overview를 제공한다.

1.1 Readership

본 문서는 다양한 독자에게 읽힐 것을 상정하고 있다. 따라서 각 부분을 서술하는 데 있어 어떠한 독자층을 상정하고 있는지를 설명한다.

1.2 Document Structure

- Introduction : 본 소프트웨어 프로젝트가 다루는 시스템의 범위에 대해 서술한다.
- System Architecture : 시스템과 각 서브시스템의 구조를 개괄적으로 기술한다.
- Database Design Requirements : 문서에서 규정된 데이터베이스 요구 사항을 기반으로, 각 데이터 엔티티의 속성과 관계를 ER diagram을 통해 표현하고 최종적으로 Relational Schema, SQL DDL를 작성한다.
- Testing Plan : Test 단계에서 component별로 점검할 항목들에 대해 기술한다.
- Development Plan : 시스템을 구현하는 데 필요한 개발 도구와 프로그래밍 언어, 라이브러리 등의 개발 환경에 대해 설명하고, 시스템 개발 일정을 기술한다.
- Index : 본 문서에서 사용된 그림, 표, 다이어그램 등의 색인을 기술한다.

2 Introduction

2.1 Objective

본 소프트웨어 프로젝트가 다루는 시스템의 범위에 대해 서술한다.

2.2 Project Scope

InteReal 프로젝트는 온라인에서 가구를 구매하려는 사용자들이 편의를 돕기 위해 고안되었다. InteReal을 통해 사용자는 거주하고 있는 방의 모습을 가상으로 구현하고, 실물 크기의 가구 모델을 배치하여 비교하고 구매 목적에 가장 부합하는 상품을 선택할 수 있다. 본인이 거주하고 있는 방을 직접 디자인하고 다양한 가구들을 서로 비교해 보면서 인테리어를 확인할 수 있도록 구성하였다.

Web System은 사용자를 확인하고, InteReal 프로그램을 배포한다. Specify Room은 사용자의 주거공간의 모양과 크기를 지정하여 Room Design을 데이터베이스에 저장한다. VR Application은 Room Design을 이용하여 사용자의 주거공간과 유사한 가상공간을 생성한다. 이렇게 생성된 공간에서 가구, 벽지, 인테리어 소품 등을 배치하고 구매할 수 있다. 데이터베이스에는 앞에서 언급한 기능들을 위한 Room Info, 인테리어 상품 정보, 구매 이력 등이 저장된다.

3 System Architecture – Overall

3.1 Objective

이번 챕터에서는 본 시스템의 전체적인 구조를 설명한다. 시스템 전체의 구조와 서브시스템 간의 관계를 서술한다.

3.2 System Organization

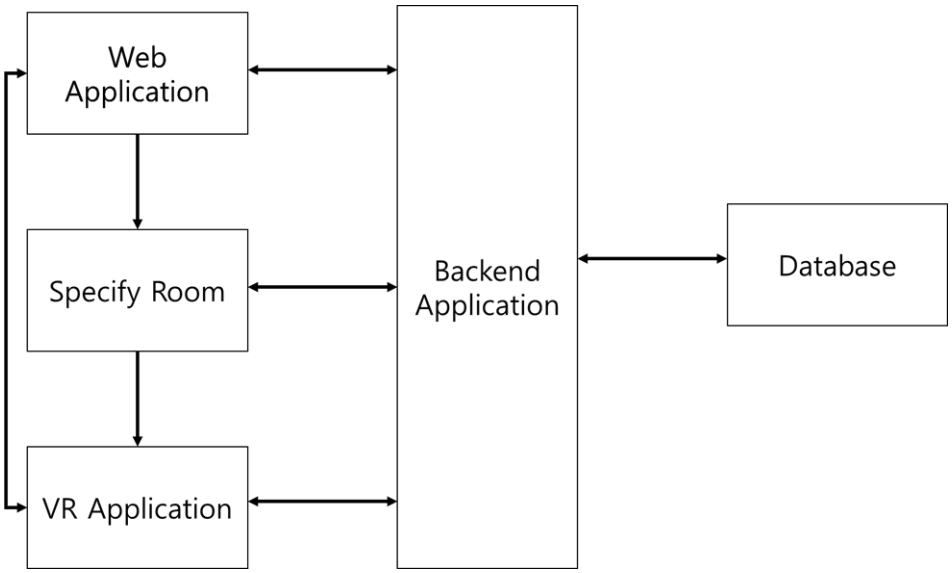


Figure 1 Overall System Architecture

본 서비스는 클라이언트-서버 모델을 참고해 설계했으며, Web Application, Specify Room, VR Application이 사용자와 상호작용하고, 이들은 Backend Application과 JSON을 기반으로 한 Web Socket 통신으로 데이터를 교환한다. Specify Room은 사용자의 주거공간을 가상공간에 옮기기 위해 모양, 크기, 창문, 문의 위치 등을 ROOM_DESIGN 자료구조로 저장하고, 이를 데이터베이스에 저장한다. VR Application에서는 ROOM_DESIGN을 이용하여, 가상공간 오브젝트 파일을 생성한다. 이 후 가상공간에서 원하는 인테리어 제품을 찾고, 생성한 방에 배치하여 데이터베이스에 저장할 수 있다. 마음에 드는 제품을 장바구니에 담거나 구매한다. Backend 서버는 다른 subsystem들과 주고받은 데이터를 데이터베이스에 반영, 혹은 저장한다.

4 Backend with Database

4.1 Objective

이번 챕터에서는 backend 시스템과 데이터베이스 간의 상호작용에 대한 구조에 대해 설명한다.

4.2 Overall Architecture

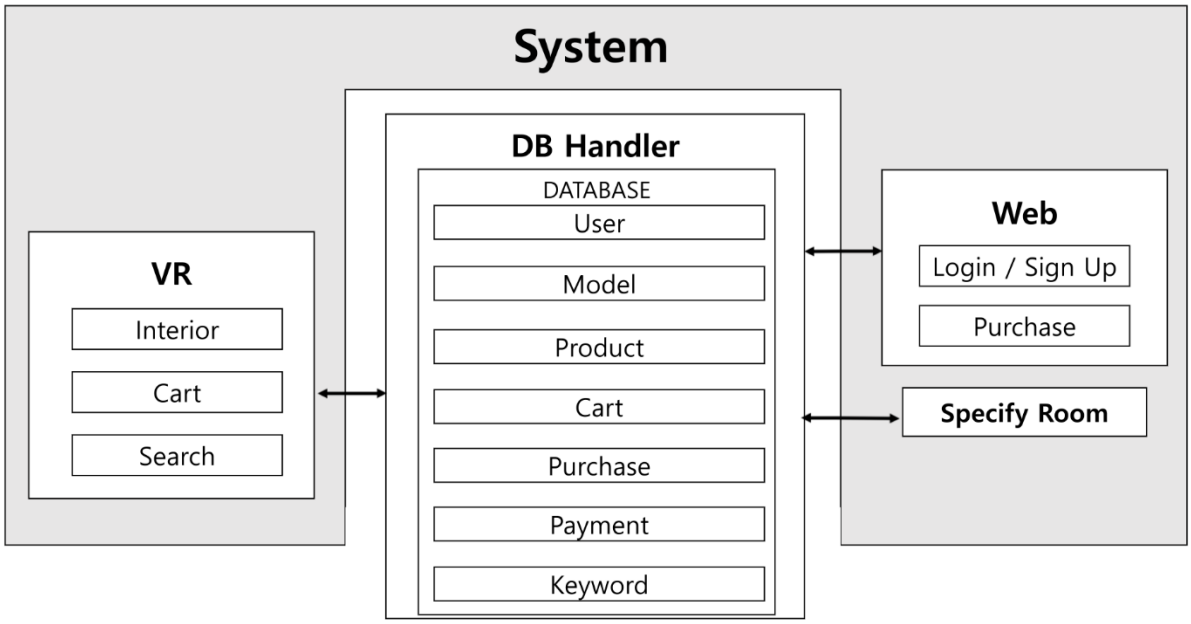


Figure 2 Backend System Architecture

Database는 Database Handler를 통해서 시스템의 전반에 걸친 System과 상호작용을 한다. 가상 공간에서 동작하는 Interior는 Database의 Model Table과, Cart 기능은 Cart Table, 그리고 Search 기능은 Product Table, Keyword Table과 Data를 주고받는다. Web에서 동작하는 Login/Sign Up 기능은 User Table과 정보를 교환한다. Specify Room은 Model Table과, 그리고 Purchase 기능은 Purchase Table, Payment Table과 정보를 교환한다.

4.3 Subcomponents

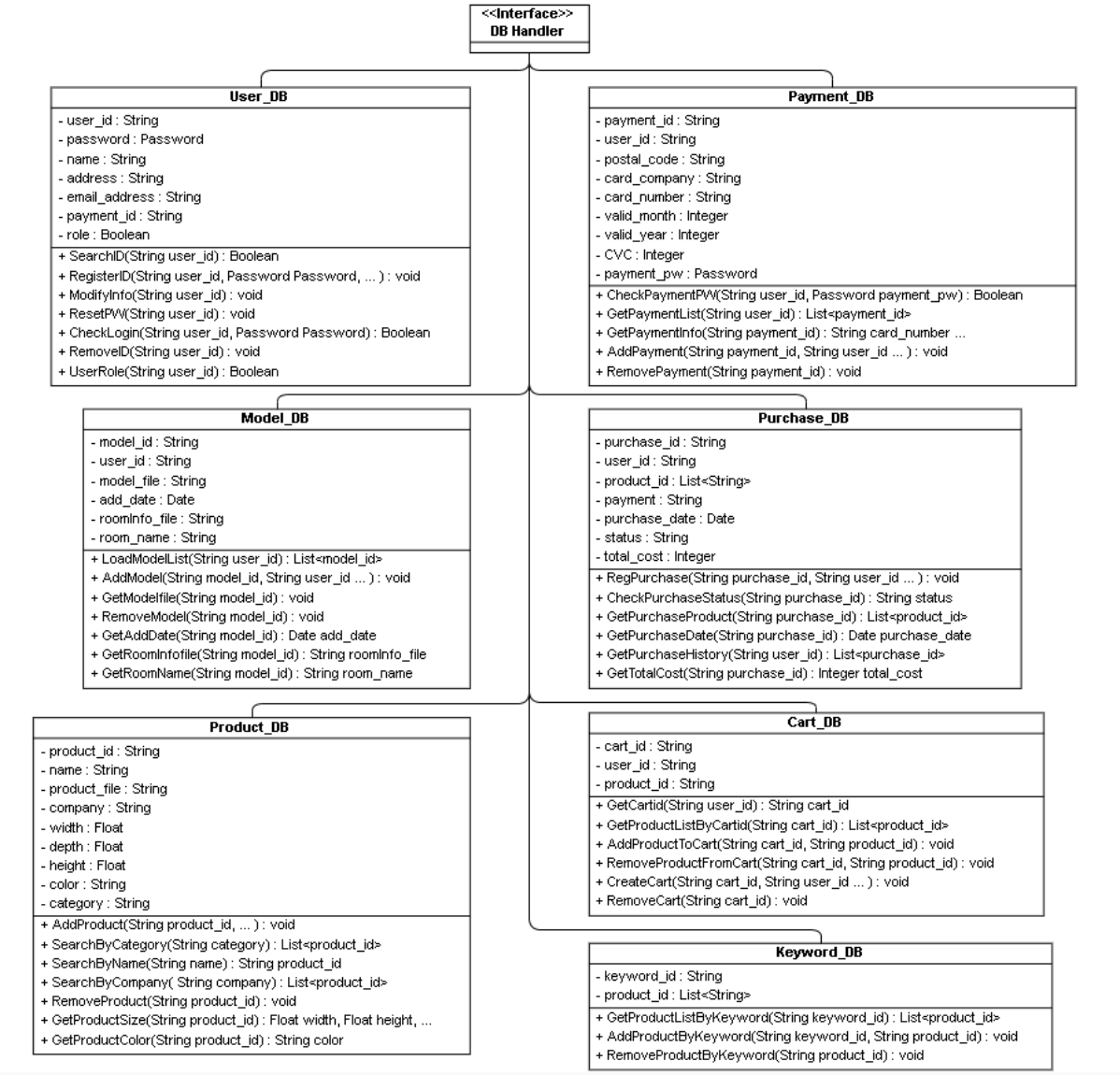


Diagram 1 Backend Sub-components

4.3.1 User_DB

a Class Diagram

User_DB
- user_id: String - password: Password - name: String - address: String - email_address: String - payment_method: String - role: Boolean
+ SearchID(String user_id): Boolean + RegisterID(String user_id, Password password, String name ...): void + ModifyInfo(String user_id): void + RestePW(String user_id): void + CheckLogin(String user_id, Password password): Boolean + RemoveID(String user_id): void + UserRole(String user_id): Boolean

Diagram 2 User DB Class Diagram

Attributes

- user_id: 사용자의 ID를 String으로 저장
- password: 사용자의 Password를 암호화된 Password 형태로 저장
- name: 사용자의 이름을 String으로 저장
- address: 사용자의 주소를 String으로 저장
- email_address: 사용자의 이메일 주소를 String으로 저장
- payment_method: 사용자의 결제수단을 String으로 저장
- role: 사용자가 소비자인지 판매자인지 정보를 Boolean으로 저장(0: 소비자 || 1: 판매자)

Methods

+ SearchID(String id): 사용자의 ID가 DB에 존재하는지를 확인하는 method
+ RegisterID(String id, Password password, String name, String address ...): 사용자의 회원가입시 DB에 추가하는 method
+ ModifyInfo(String id): 사용자의 정보를 수정하는 method
+ ResetPW(String id): 사용자가 비밀번호를 잊었을 경우 비밀번호를 초기화하는 method
+ CheckLogin(String id, Password password): 사용자가 ID와 PW로 로그인하는 method

+ RemoveId(String id): 사용자가 탈퇴할 경우 정보를 삭제하는 method
+ UserRole(String id): 사용자가 소비자인지 판매자인지의 정보를 알려주는 method

b Sequence Diagram

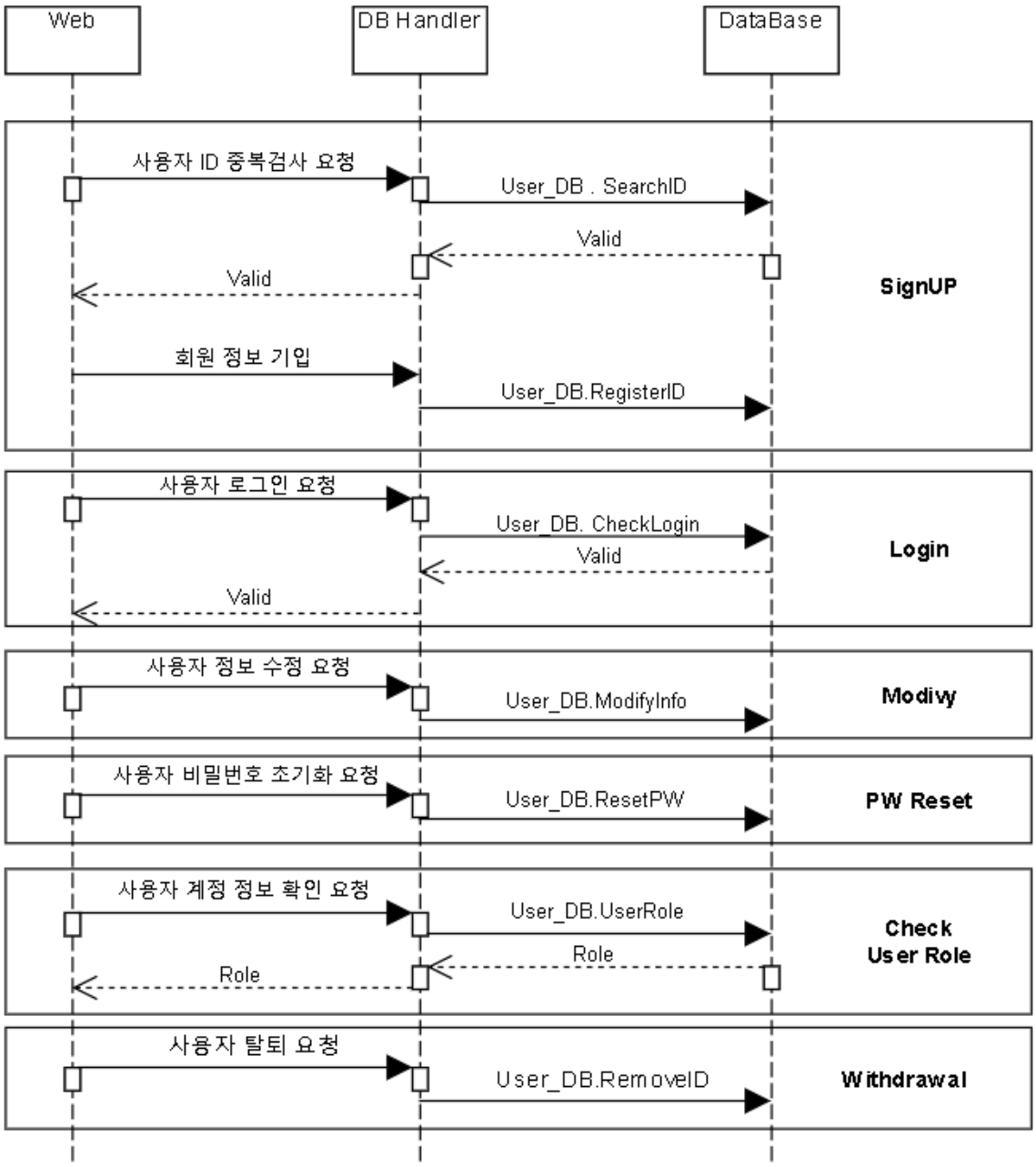


Diagram 3 User DB Sequence Diagram

4.3.2 Model_DB

a Class Diagram

Model_DB
- model_id: String - user_id: String - model_file: String - add_date: Date - roomInfo_file: String - room_name: String
+ LoadModelList(String user_id): List<model_id> + AddModel(String model_id, String user_id ...): void + GetModelfile(String model_id): String model_file + RemoveModel(String model_id): void + GetAddDate(String model_id): Date add_date + GetRoomInfofile(String model_id): String roomInfo_file + GetRoomName(String model_id): String room_name

Diagram 4 Model DB Class Diagram

Attributes

- model_id: 사용자가 생성한 모델에 부여하는 식별코드
- user_id: 사용자의 ID
- model_file: 모델의 파일 이름
- add_date: 모델이 생성된 날짜
- roomInfo_file: 모델의 배치정보를 갖고 있는 파일 이름
- room_name: 모델의 이름

Methods

+ LoadModelList(String user_id): 사용자가 생성한 모델의 리스트를 알려주는 method
+ AddModel(String model_id, String user_id, String model_file, Date add_date): 사용자가 모델을 새로 생성했을 때 모델을 등록하는 method
+ GetModelfile(String model_id): 선택한 모델의 파일명을 반환하는 method
+ RemoveModel(String model_id): 사용자가 모델을 삭제했을 때 DB에서 제거하는 method
+ GetAddDate(String model_id): 모델을 생성한 날짜를 반환하는 method
+ GetRoomInfofile(String model_id): 선택한 모델의 RoomInfo 파일명을 반환하는 method
+ GetRoomName(String model_id): 선택한 모델의 방 이름을 반환하는 method

b Sequence Diagram

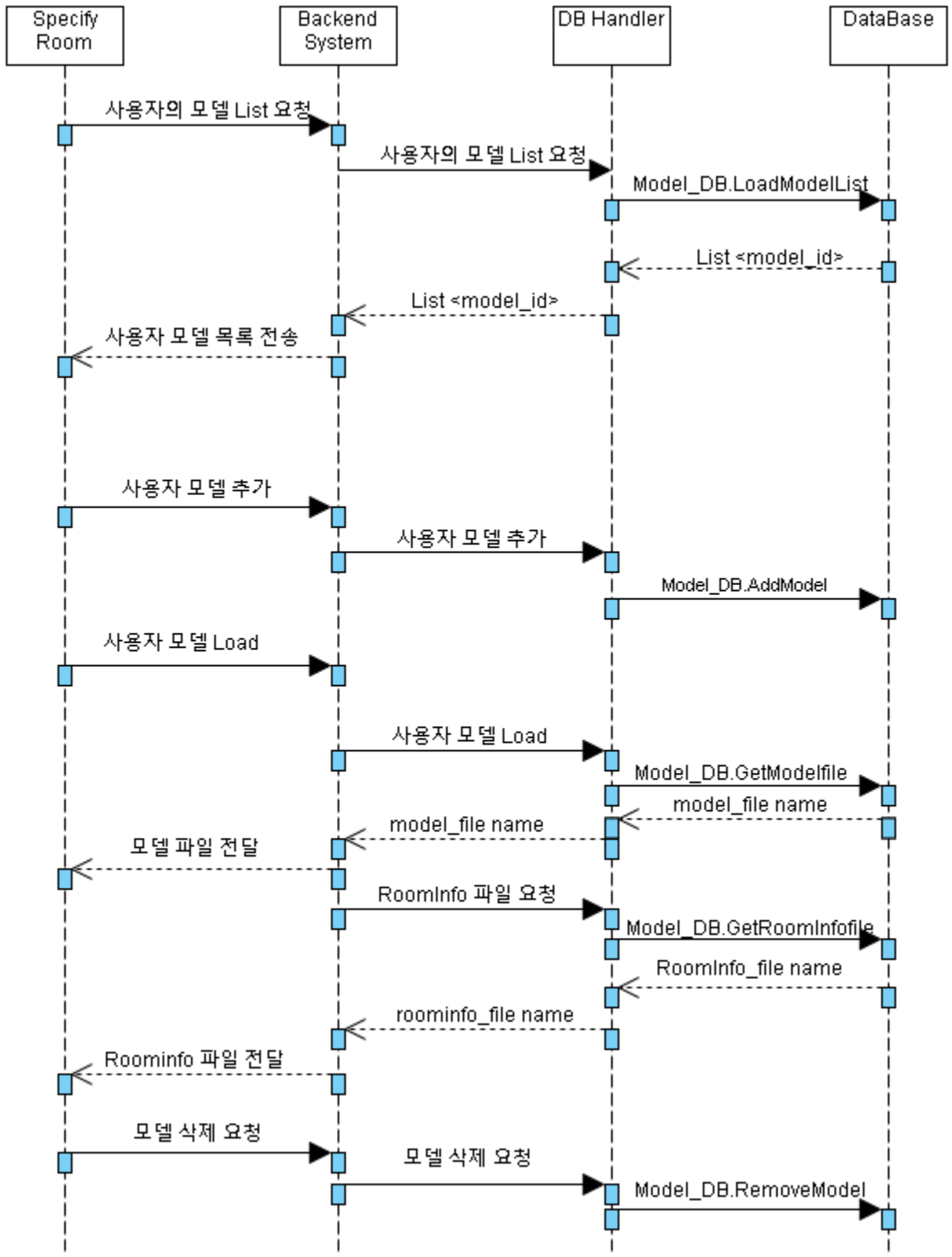


Diagram 5 Model DB Sequence Diagram

4.3.3 Product_DB

a Class Diagram

Product_DB
- product_id: String - name: String - product_file: String - company: String - width: Float - depth: Float - height: Float - color: String - category: String
+ AddProduct(String product_id , ...): void + SearchByCategory(String category): List<product_id> + SearchByName(String name): String product_id + SearchByCompany(String company): List<product_id> + RemoveProduct(String product_id): void + GetProductSize(String product_id): Float width, Float height ... + GetProductColor(String product_id): String + GetProductfile(String product_id): String product_file

Diagram 6 Product DB Class Diagram

`Attributes

- product_id: 상품에 대한 고유식별코드
- name: 상품의 이름에 대한 정보
- product_file: 상품의 모델 파일의 이름
- company: 상품의 회사정보
- width: 상품의 가로 사이즈
- height: 상품의 세로 사이즈
- depth: 상품의 높이 사이즈
- color: 상품의 색상 정보
- category: 상품의 카테고리

Methods

+ AddProduct(String product_id, String name, String product_file, ...): 새로운 상품이 추가되었을 때 이를 DB에 등록하는 method
+ SearchByCategory(String category): 카테고리로 상품을 분류하였을 때 해당 상품의 List를 반환하는 method
+ SearchByName(String name): 상품의 이름으로 상품을 검색하였을 때 상품을 알려주는 method
+ SearchByCompany(String company): 회사로 상품을 찾고 싶을 때, 특정 회사 상품의 List를 반환해주는 method
+ RemoveProduct(String product_id): 상품의 정보를 제거하는 method
+ GetProductSize(STring product_id): 상품의 크기를 반환해주는 method
+ GetProductColor(String product_id): 상품의 색깔 정보를 반환해주는 method
+ GetProductfile(String product_id): 상품의 파일명을 반환해주는 method

b Sequence Diagram

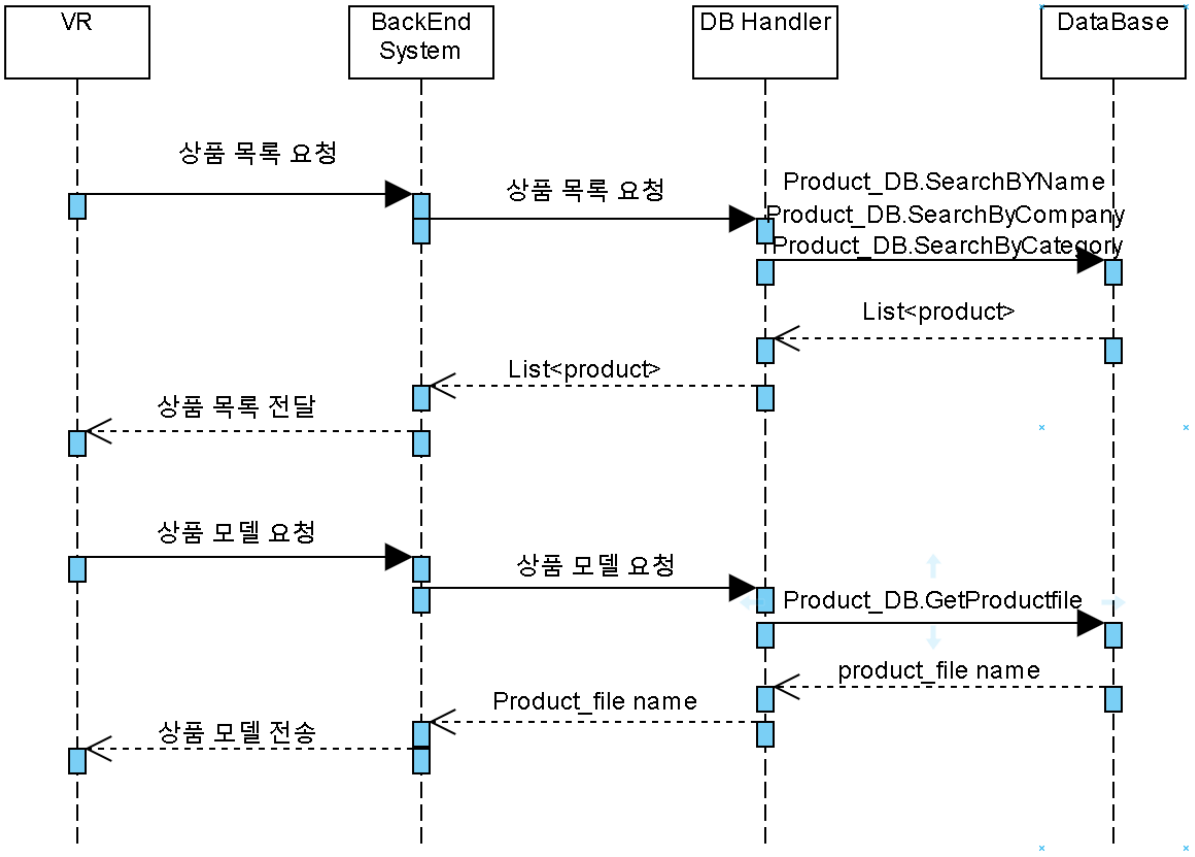


Diagram 7 Product DB Sequence Diagram

4.3.4 Cart_DB

a Class Diagram

Cart_DB
- cart_id: String - user_id: String - product_id: List<String>
+ GetCartid(String user_id): String cart_id + GetProductListByCartid(String cart_id): List<product_id> + AddProductToCart(String cart_id, String product_id): void + RemoveProductFromCart(String cart_id, String product_id): void + CreateCart(String cart_id, String user_id ...): void + RemoveCart(String cart_id): void

Diagram 8 Cart DB Class Diagram

Attributes

- cart_id: 장바구니의 식별코드
- user_id: 사용자의 id
- product_id: 장바구니에 담겨있는 상품들의 목록

Methods

+ GetCartid(String user_id, String model_id): 사용자의 id와 모델 정보를 통해 장바구니 코드를 반환하는 method
+ GetProductListByCartid(String cart_id): 특정 장바구니에 담겨있는 상품 List를 반환하는 method
+ AddProductToCart(String cart_id, String product_id): 장바구니에 특정 상품을 추가하는 method
+ RemoveProductFromCart(String cart_id, String product_id): 장바구니에서 특정 상품을 제거할 때 동작하는 method
+ CreatCart(String cart_id, String user_id, String product_id): 새로운 장바구니를 생성할 때 동작하는 method
+ RemoveCart(String cart_id): 장바구니를 전체 삭제할 경우 동작하는 method

b Sequence Diagram

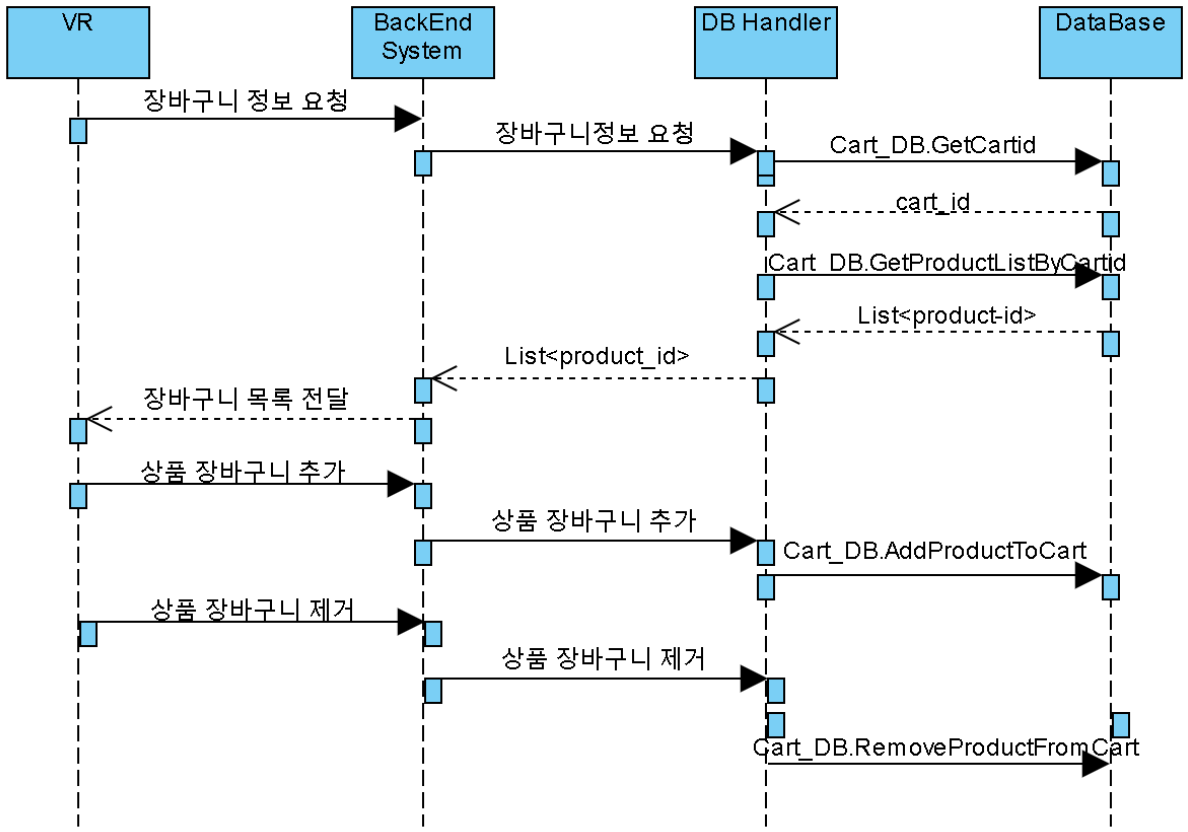


Diagram 9 Cart DB Sequence Diagram

4.3.5 Purchase_DB

a Class Diagram

Purchase_DB
- purchase_id: String - user_id: String - product_id: List<String> - payment: String - purchase_date: Date - status: String - total_cost: Integer
+ RegPurchase(String purchase_id, String user_id ...): void + CheckPurchaseStatus(String purchase_id): String status + GetPurchaseProduct(String purchase_id): List<product_id> + GetPurchaseDate(String purchase_id): Date purchase_date + GetPurchaseHistory(String user_id): List<purchase_id> + GetTotalCost(String purchase_id): Integer total_cost

Diagram 10 Purchase DB Class Diagram

Attributes

- purchase_id: 구매정보에 대한 식별코드
- user_id: 사용자의 id
- product_id: 구매한 상품의 id
- payment: 결제방법
- purchase_date: 구매 날짜
- status: 제품의 배송 상태 (+ 반품/환불)
- cost: 상품의 가격

Methods

+ RegPurchase(String purchase_id, String user_id, String product_id, String payment ...): 상품이 결제되었을 때, 결제 정보를 등록하는 method
+ CheckPurchaseStatus(String purchase_id): 구매상품의 배송 상태를 확인하는 method
+ GetPurchaseProduct(String purchase_id): 상품 구매 목록을 알려주는 method
+ GetPurchaseDate(String purchase_id): 상품 구매 날짜를 알려주는 method
+ GetPurchaseHistory(String user_id): 사용자의 구매 내역 List를 알려주는 method

+ GetTotalCost(String purchase_id): 특정 구매 내역의 총 가격을 알려주는 method

b Sequence Diagram

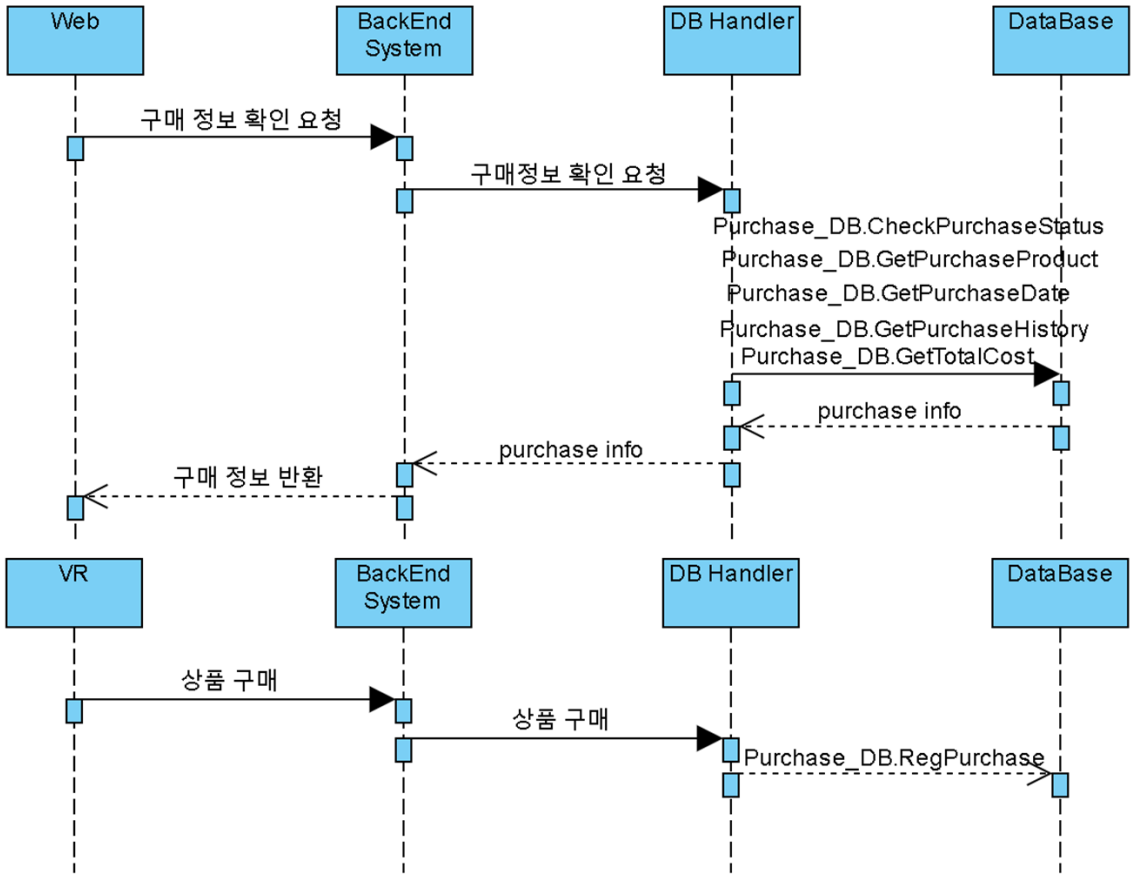


Diagram 11 Purchase DB Sequence Diagram

4.3.6 Payment_DB

a Class Diagram

Payment_DB
- payment_id: String - user_id: String - postal_code: String - card_company: String - card_number: String - valid_month: Integer - valid_year: Integer - CVC: Integer - payment_pw: Password
+ CheckPaymentPW(String user_id, Password payment_pw): Boolean + GetPaymentList(String user_id): List<payment_id> + GetPaymentInfo(String payment_id): String card_number ... + AddPayment(String payment_id, String user_id ...): void + RemovePayment(String payment_id): void

Diagram 12 Payment DB Class Diagram

Attributes

- payment_id: 결제 정보에 대한 식별코드
- user_id: 사용자의 id
- postal_code: 우편 배송 코드
- card_company: 카드 회사
- card_number: 결제 카드 번호
- valid_month: 카드 유효기간(월)
- valid_year: 카드 유효기간(년)
- cvc: 카드 cvc
- payment_pw: 결제 비밀번호

Methods

+ CheckPaymentPW(String user_id, Password payment_pw): 사용자의 결제 비밀번호의 valid 여부를 확인하는 method
+ GetPaymentList(String user_id): 사용자의 결제 수단의 목록을 나타내는 method

+ GetPaymentInfo(String payment_id): 결제수단의 정보를 나타내는 method
+ AddPayment(String payment_id, String user_id ...): 사용자의 결제수단을 추가하는 method
+ RemovePayment(String payment_id): 사용자의 결제수단을 제거하는 method

b Sequence Diagram

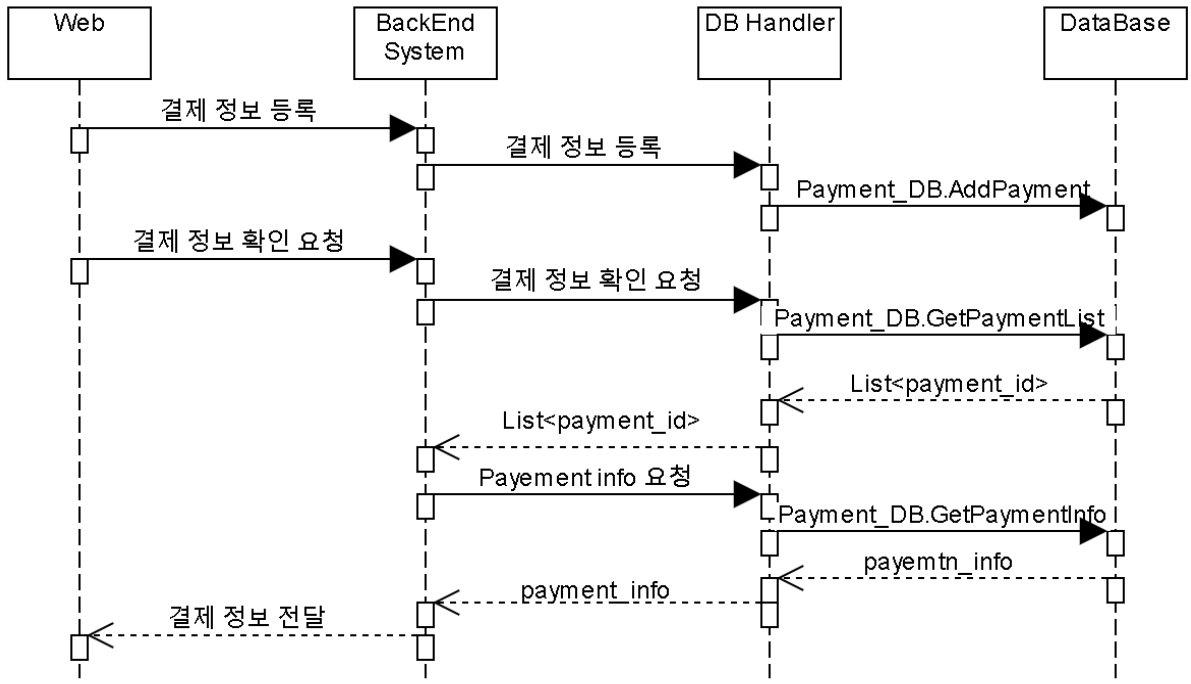


Diagram 13 Payment DB Sequence Diagram

4.3.7 Keyword_DB

a Class diagram

Keyword_DB
- keyword_id: String - product_id: List<String>
+ GetProductListByKeyword(String keyword_id): List <product_id> + AddProductByKeyword(String keyword_id, String product_id): void + RemoveProductByKeyword(String product_id): void

Diagram 14 Keyword DB Class Diagram

Attributes

- keyword_id: 키워드 명
- product_id: 키워드에 해당하는 상품의 id

Methods

+ GetProductListByKeyword(String keyword_id): 키워드에 부합하는 제품의 목록을 반환하는 method
+ AddProductByKeyword(String keyword_id, String product_id): 새로운 상품을 키워드에 등록하는 method
+ RemoveProductByKeyword(String product_id): 상품을 키워드로부터 제거하는 method

b Sequence Diagram

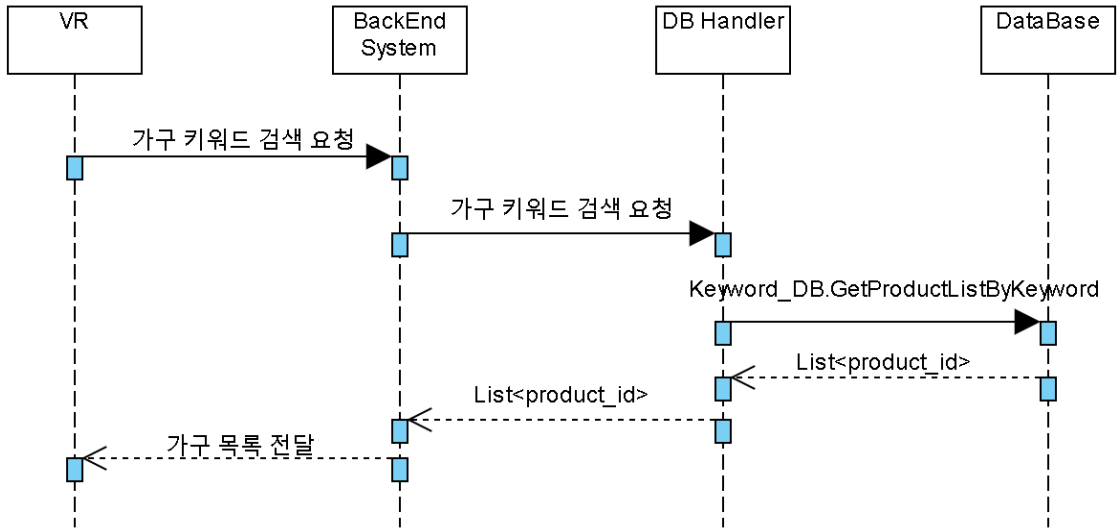


Diagram 15 Keyword DB Sequence Diagram

5 Web Application

5.1 Objective

이 파트에서는 웹 시스템의 각 sub-component가 user 및 backend server와 어떤 상호작용을 하고, 그 과정에서 어떠한 정보를 주고받는지 설명한다.

5.2 Overall Architecture

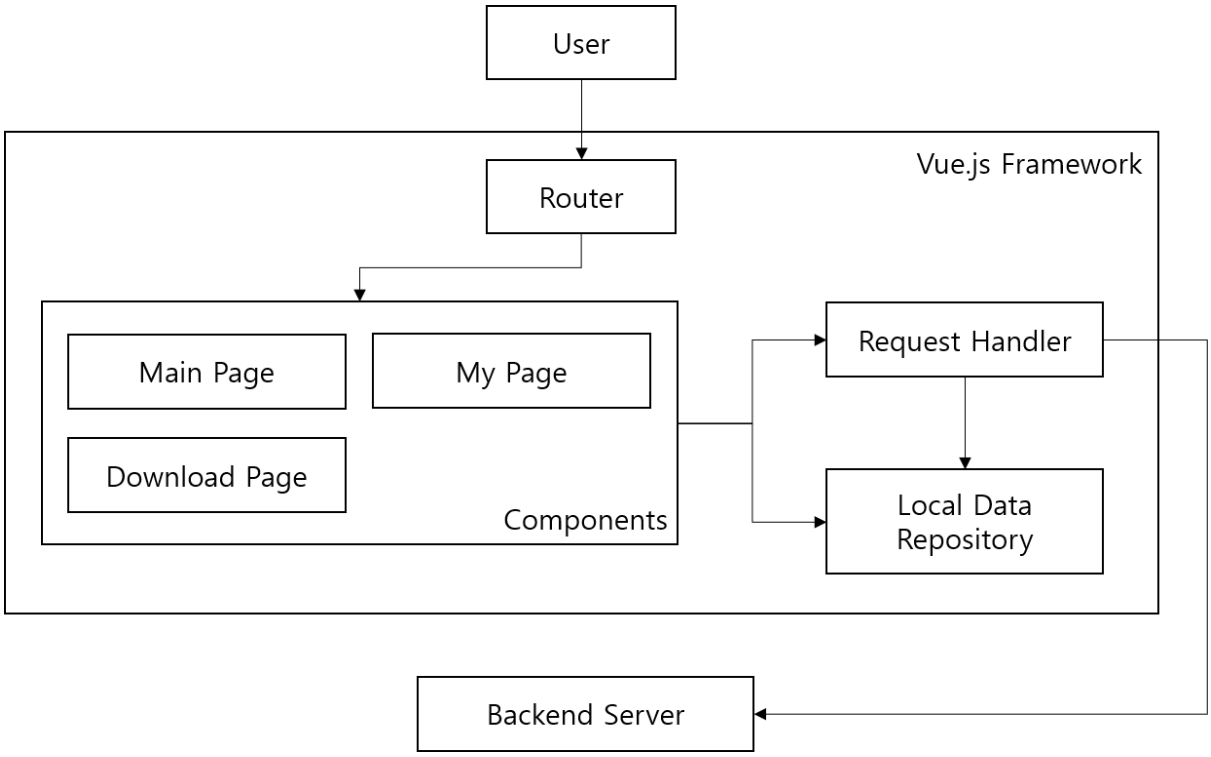


Figure 3 Web Application Architecture

웹 시스템은 위와 같은 구조로 이루어져 있다. 사용자는 Veu.js의 프레임워크 상에서 작동하는 웹 어플리케이션과 다양한 상호작용을 한다. Router는 사용자의 입력 주소에 따라 그에 맞는 sub-component를 사용자가 이용할 수 있도록 한다. 사용자가 입력한 정보들은 해당 sub-component는 Request Handler를 통해 Backend Server와 통신해 필요한 정보는 주고받는다. sub-component로는 Main Page, My Page, Download Page가 있다.

5.3 Subcomponents

5.3.1 Main Page

a Class Diagram

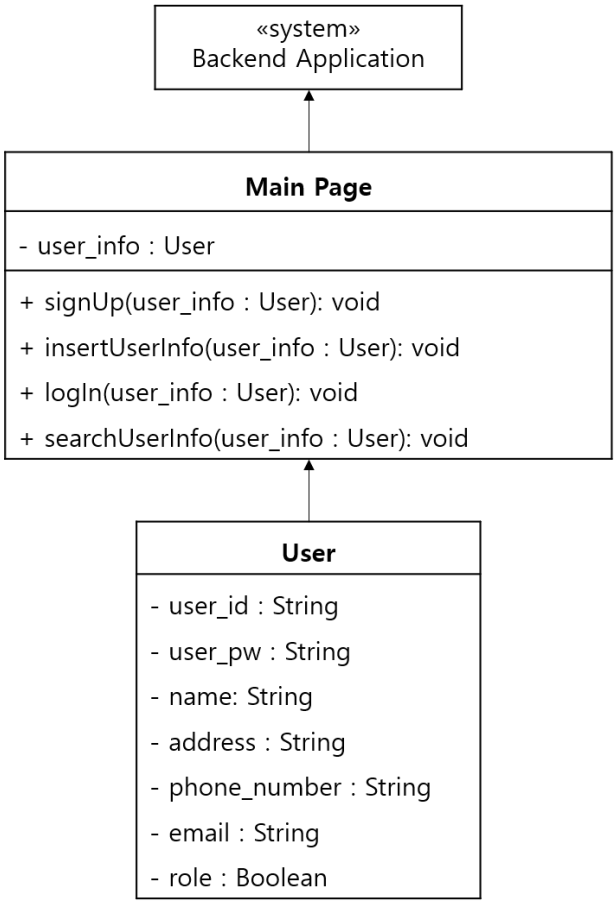


Diagram 16 Main Page Class Diagram

i Main Page

Attributes

- user_info: sign up 혹은 log in을 위한 사용자 정보를 담아두는 공간

Methods

+ signUp(user_info: User): 사용자가 회원가입을 위한 정보를 입력, Main Page로 전송하는 함수
+ insertUserInfo(user_info: User): 입력된 정보를 backend 서버로 전송해 회원으로 등록하는 함수
+ logIn(user_info: User): 사용자가 로그인을 위해 id, pw를 입력, Main Page로 전송하는 함수

+ searchUserInfo(user_info: User): 로그인을 시도하는 사용자가 등록된 회원인지 확인하는 함수

ii User

Attributes

- user_id: 사용자 id
- user_pw: 사용자 password
- name: 사용자의 이름
- address: 사용자의 주소
- phone_number: 사용자 연락처
- email: 사용자 메일 주소
- role: 사용자가 판매처인지를 나타내는 값. 판매자이면 True, 일반 사용자이면 False.

b Sequence Diagram

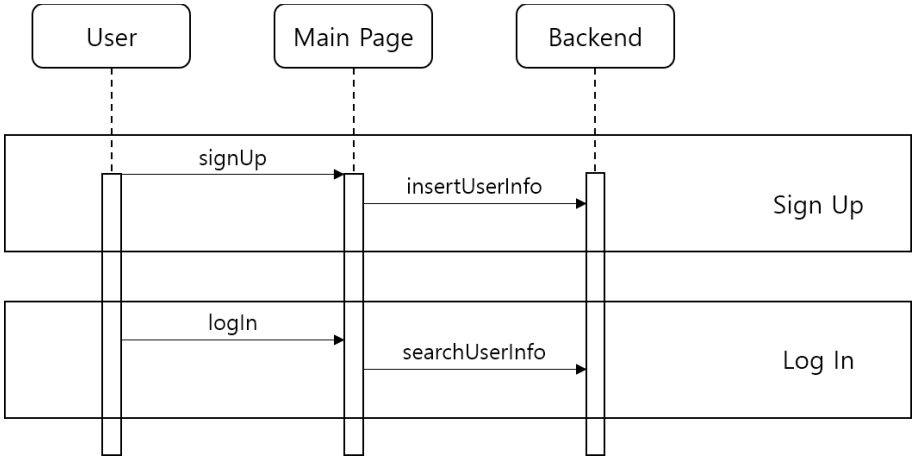


Diagram 17 Main Page Sequence Diagram

5.3.2 My Page

a Class diagram

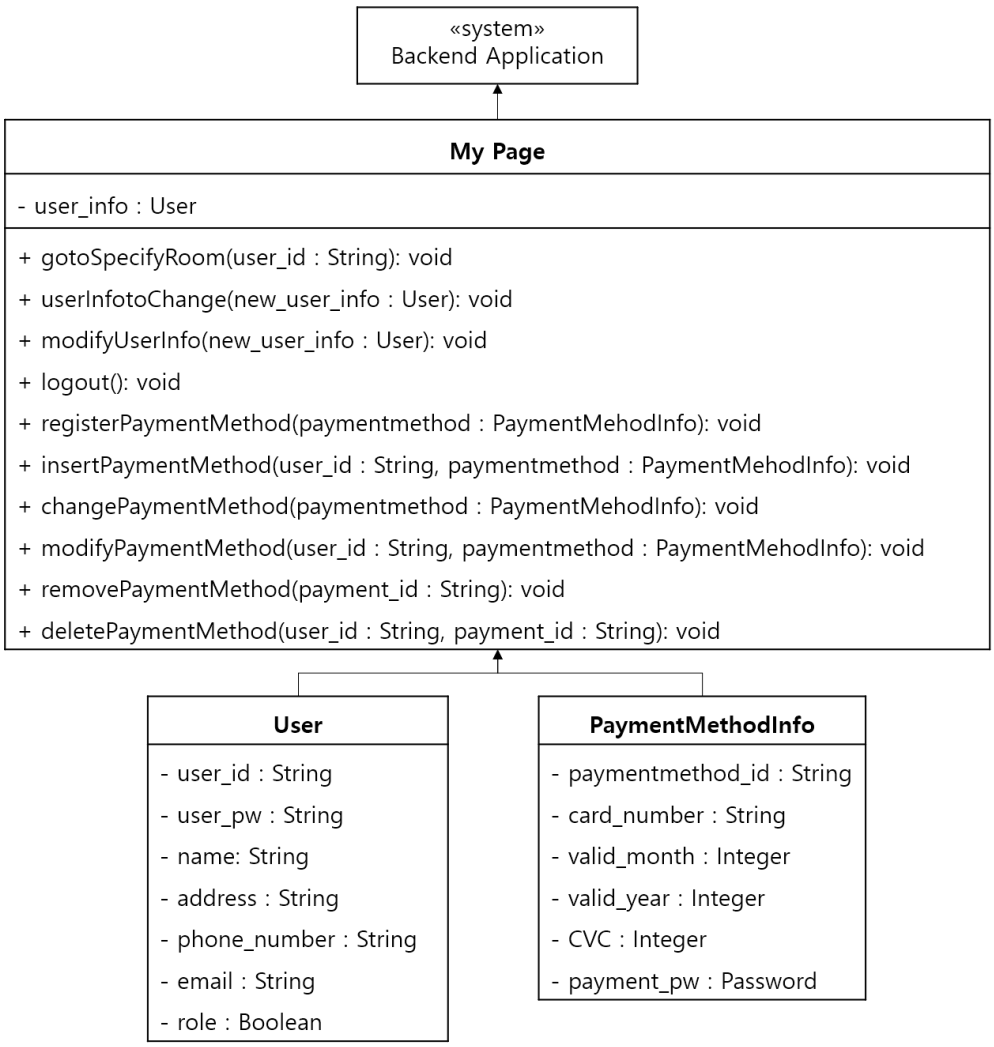


Diagram 18 My Page Class Diagram

i My Page

Attributes

- user_info: 로그인한 사용자의 정보
- saved_room_id: 사용자가 그동안 저장했던 방 모델의 id

Methods

+ gotoSpecifyRoom(user_id: String): 사용자의 id 를 넘기며 SpecifyRoom 으로 이동한다.
+ userInfotoChange(new_user_info: User): 사용자가 수정하고자 하는 정보를 입력 받는다.

+ modifyUserInfo(new_user_info: User): 입력한 정보를 토대로 사용자의 정보를 수정한다.
+ logout(): 로그아웃한 뒤 Main Page 로 돌아간다.
+ registerPaymentMethod(paymentmethod : PaymentMehodInfo): 사용할 결제 수단 정보를 입력한다.
+ insertPaymentMethod(user_id : String, paymentmethod : PaymentMehodInfo): 입력 받은 정보를 바탕으로 새로운 결제 수단을 등록한다.
+ changePaymentMethod(paymentmethod : PaymentMehodInfo): 수정하고자 하는 결제 수단 정보를 입력한다.
+ modifyPaymentMethod(user_id : String, paymentmethod : PaymentMehodInfo): 입력 받은 정보를 바탕으로 기존의 결제 수단을 수정한다.
+ removePaymentMethod(payment_id : String): 삭제하고자 하는 결제 수단을 선택한다.
+ deletePaymentMethod(user_id : String, payment_id : String): 선택된 결제 수단 정보를 삭제한다.

ii User

Attributes

- user_id: 사용자 id
- user_pw: 사용자 password
- name: 사용자의 이름
- address: 사용자의 주소
- phone_number: 사용자 연락처
- email: 사용자 메일 주소
- role: 사용자가 판매자인지를 나타내는 값. 판매자이면 True, 일반 사용자이면 False값을 가진다.

iii PaymentMethodInfo

Attributes

- paymentmethod_id: 결제 수단의 id
- card_number: 카드 번호 16자리
- valid_month: 카드의 유효 월
- valid_year: 카드의 유효 년도
- CVC: 카드의 CVC 숫자
- payment_pw 카드 비밀번호

b Sequence Diagram

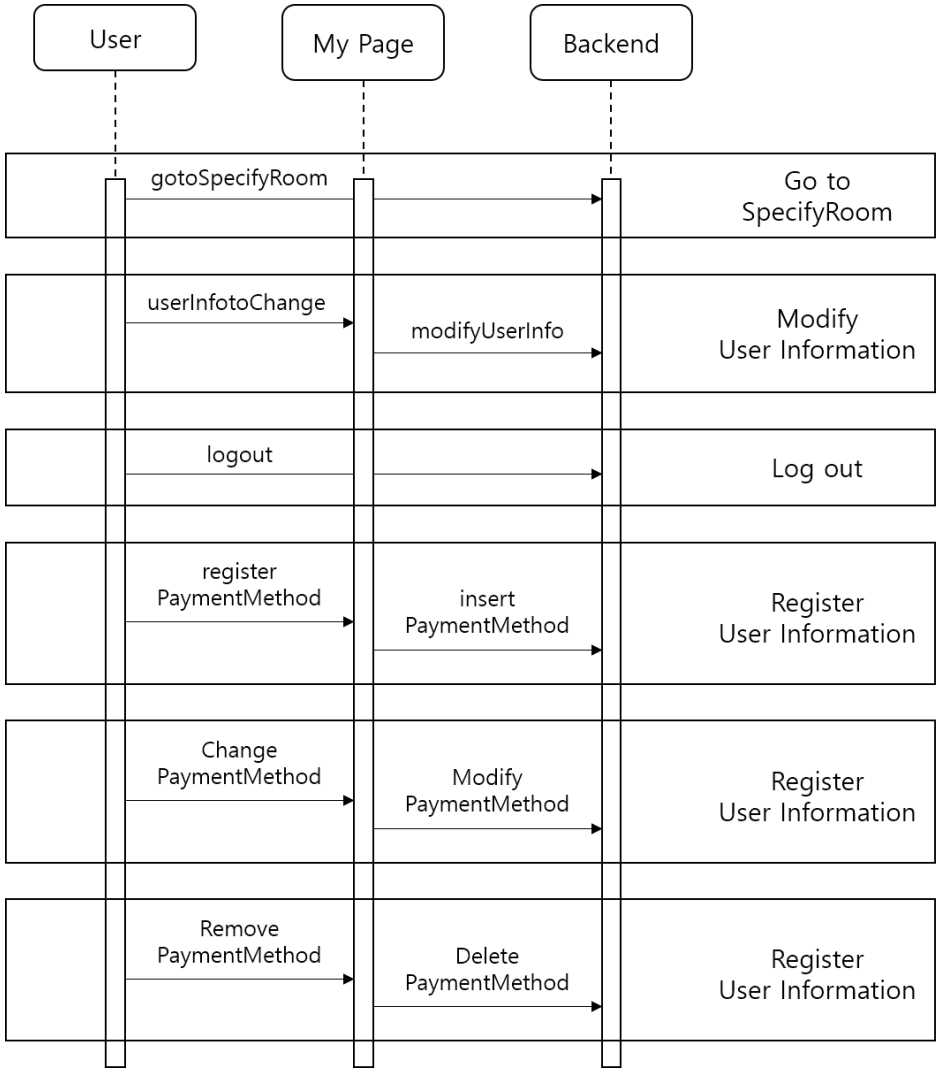


Diagram 19 My Page Sequence Diagram

5.3.3 Download Page

a Class Diagram

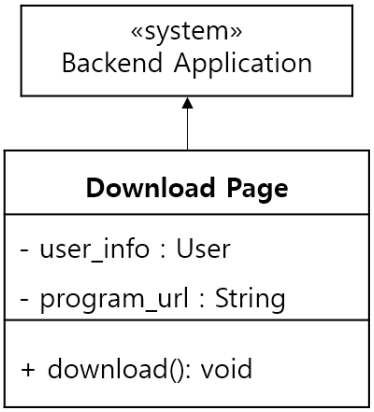


Diagram 20 Download Page Class Diagram

i Download Page

Attributes

- user_info: 로그인한 사용자의 정보
- program_url: 다운받을 프로그램의 주소

Methods

+ download(): InteReal 실행을 위한 프로그램 다운로드

b Sequence Diagram

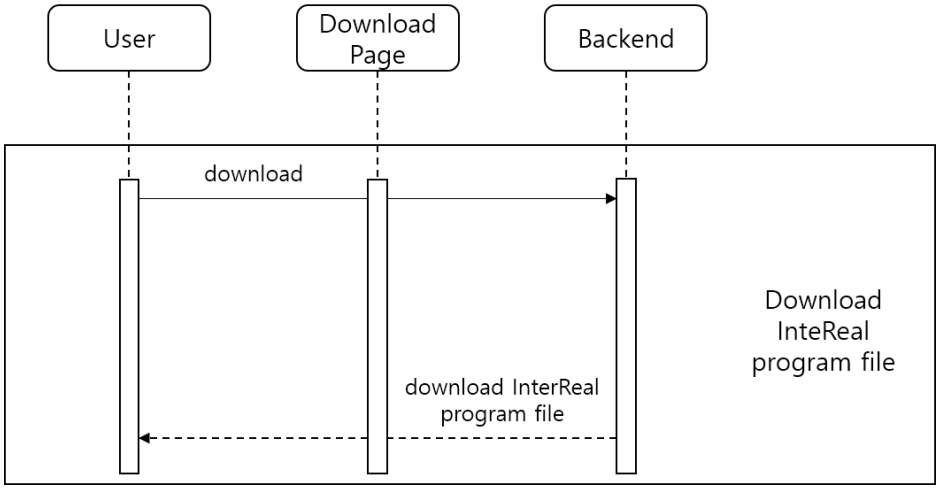


Diagram 21 Download Page Sequence Diagram

6 Room Specification

6.1 Objective

전체 시스템 아키텍처 중 사용자의 주거공간의 모델링을 시스템의 구조와 각 컴포넌트의 구성, 컴포넌트 간의 관계를 서술한다.

6.2 Overall Architecture

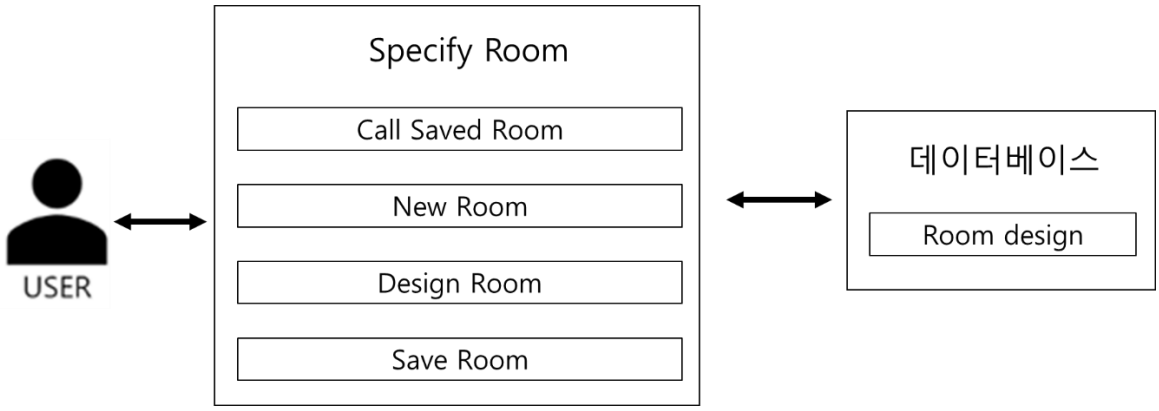


Figure 4 Room Specification Architecture

Specify Room 시스템의 구조는 위와 같다. 사용자의 Call Saved Room을 통해 데이터베이스에 미리 저장된 Room Design을 불러오거나 New Room을 통해 새로운 Room Design을 만들 수 있다. Design Room을 통해 새로운 Room Design을 지정할 수 있고, 지정한 Room Design을 SaveRoom을 통해 데이터베이스에 저장할 수 있다. 각각의 시스템이 user와 상호작용하고, Call Saved Room, Save Room은 데이터베이스와 상호작용한다.

6.3 Subcomponents

6.3.1 Specify Room

a Class Diagram

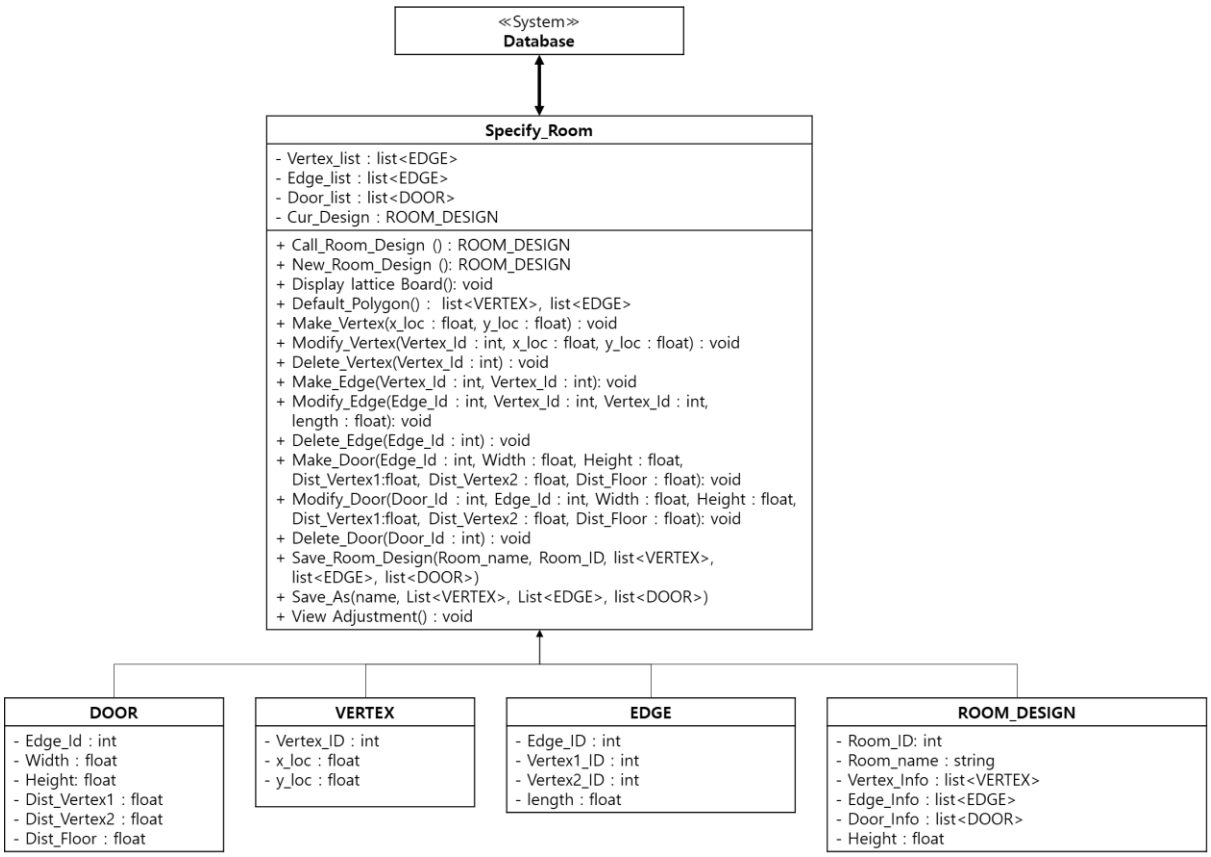


Diagram 22 Specify Room Class Diagram

i Specify_Room

Attributes

- Vertex_list: 모든 vertex의 정보를 저장
- Edge_list: 모든 edge의 정보를 저장
- Door_list: 모든 door의 정보를 저장하기 위한 list
- Cur_design: 현재의 room design의 정보를 저장한다.

Methods

+ Call_Room_Design(): 저장된 방 정보 중 하나를 선택하여 불러온다.

+ New_Room_Design(): Room ID를 생성하고, Room name을 입력으로 받아서, ROOM_DESIGN 자료구조에 저장한다.
+ Display_Lattice_Board(): 방의 모양과 크기를 지정해주기 위해 vertex와 edge를 지정해줄 수 있는 격자를 화면에 출력한다.
+ Default_Polygon(): 방의 정보를 편리하게 지정할 수 있게, 길이가 1인 정 N각형의 도형을 격자를 출력한다.
+ Make_Vertex(x_loc: float, y_loc: float): 격자 위의 선택한 위치에 점을 찍는다. Vertex_Id를 생성하고, vertex_Id, 선택한 위치의 x좌표, y좌표를 Vertex자료구조로 저장하고, list<VERTEX>에 추가한다.
+ Modify_Vertex(Vertex_Id: int, x_loc: float, y_loc: float): 선택한 vertex를 좌표 또는 마우스 클릭을 통해 옮길 수 있게 한다. 해당 ID를 가진 vertex를 list<VERTEX>에서 찾아 수정한다.
+ Delete_Vertex(Vertex_Id): 선택한 vertex를 격자 위에서 삭제하고, 해당 ID를 가진 vertex를 list<VERTEX>에서 삭제한다.
+ Make_Edge(Vertex_Id, Vertex_Id): Vertex 사이를 잇는 edge를 격자 위에 그려준다. Edge_ID를 생성하고, 두개의 vertex의 Id와 그 사이의 길이를 입력으로 받아 EDGE 자료구조로 저장하고, list<EDGE>에 추가한다.
+ Modify_Edge(Edge_Id: int, Vertex_Id: int, Vertex_Id: int, length: float): 선택한 edge를 길이를 지정하여 길이를 변경한다. 해당 ID를 가진 edge를 list<EDGE>에서 찾아 수정한다.
+ Delete_Edge(Edge_Id): 선택한 edge를 격자 위에서 삭제하고, 해당 ID를 가진 edge를 list<EDGE>에서 삭제한다.
+ Make_Door(Edge_Id: int, Width: float, Height: float, Dist_Vertex1: float, Dist_Vertex2: float, Dist_Floor: float): Door_ID를 생성한다. 선택한 edge위에서 문을 만든다. 선택된 Edge에서 포함되어 있는 두 Vertex 중 하나의 Vertex와의 거리와 문의 width, height를 입력함으로써 문의 가로폭과 세로폭을 지정할 수 있다. 창문의 아랫부분과 바닥까지의 길이 dist_floor를 입력 받는다. 이러한 정보들을 DOOR 자료구조로 저장하고, list<DOOR>에 추가한다.
+ Modify_Door(Door_Id: int, Edge_Id: int, Width: float, Height: float, Dist_Vertex1: float, Dist_Vertex2: float, Dist_Floor: float): 선택한 door의 가로, 세로폭과 높이, 위치를 변경한다. 해당 ID를 가진 door를 list<DOOR>에서 찾아 수정한다.
+ Delete_Door(Door_Id: int): 선택한 door를 격자 위에서 삭제하고, 해당 ID를 가진 door를 list<DOOR>에서 삭제한다.
+ Save_Room_Design(Room_name, Room_ID, list<VERTEX>, list<EDGE>, list<DOOR>): 현재 사용중인 list<VERTEX>, list<EDGE>, list<DOOR>를 Room_Design에 입력하고, Room_Design을 데이터 베이스에 저장한다.
+ Save_As(Room_name, list<Vertex>, list<Edge>, list<DOOR>): 새로운 Room_ID를 생성, room name을 새로 입력 받아 새로운 ROOM_DESIGN을 생성한다. 현재 사용중인 list<VERTEX>, list<VERTEX>, list<DOOR>를 ROOM_DESIGN에 입력하고, ROOM_DESIGN를 데이터 베이스에

저장한다.

+ View_Adjustment(): 화면에 출력되는 격자의 크기를 조절할 수 있다.

ii Door

Attributes

- Edge_ID: door가 존재하는 edge ID
- Width: 문의 가로 폭
- Height: 문의 세로 폭
- Dist_Vertex1: vertex1에서의 거리
- Dist_Vertex2: vertex2에서의 거리
- Dist_Floor: 바닥에서부터 창문 하단의 거리

iii Vertex

Attributes

- Vertex_ID: Vertex의 ID number
- x_loc: 해당 vertex의 x좌표
- y_loc: 해당 vertex의 y좌표

iv Edge

Attributes

- Edge_ID: edge의 ID number
- Vertex1_ID: edge를 구성하는 vertex1의 ID number
- Vertex2_ID: edge를 구성하는 vertex2의 ID number
- length: edge의 길이

v Room_Design

Attributes

- Room_ID: ROOM_DESIGN의 ID number
- Room_name: 사용자가 지정한 방의 이름
- Vertex_Info: 사용하고 있는 전체 VERTEX list
- Edge_Info: 사용하고 있는 전체 Edge list

- Door_Info: 사용하고 있는 전체 DOOR list
- Height: 방의 바닥과 천장 사이의 거리

b Sequence Diagram

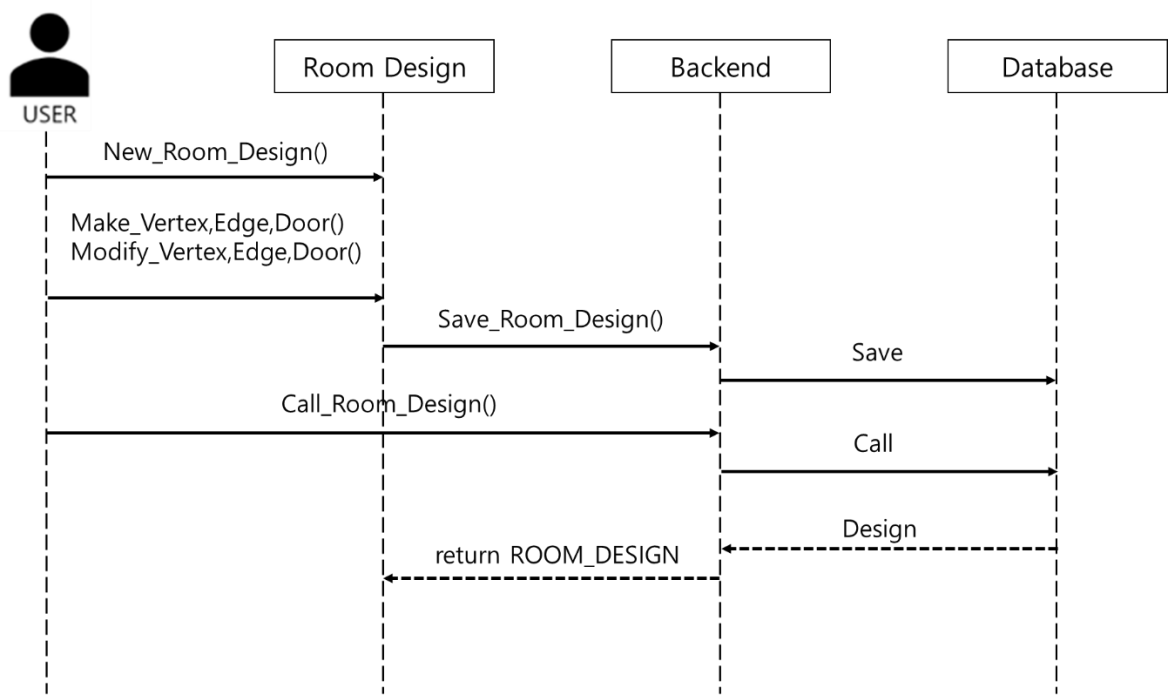


Diagram 23 Specify Room Sequence Diagram

7 Virtual Reality

7.1 Objective

이 파트에서는 VR system의 각 subsystem이 어떻게 user와 상호작용하는지, 그리고 database나 server와 같은 backend와는 어떤 정보를 주고받는지 그 구조를 설명한다.

7.2 Overall Architecture

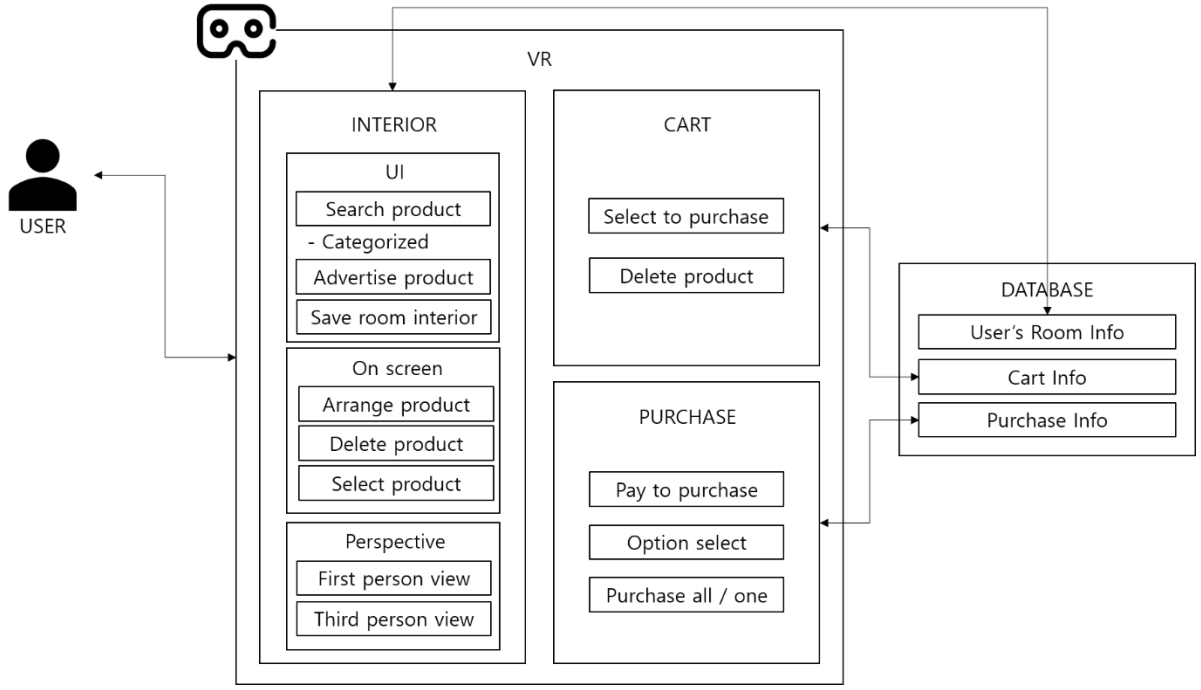


Figure 5 VR System Architecture

VR 시스템의 구조는 위와 같은데, user의 input을 받아 그 input에 따라 행동하며, 어떠한 output을 내기 위해 필요한 정보를 database에서 가져온다. 크게 interior subsystem, cart subsystem, purchase subsystem으로 이루어져 있으며, 각각의 system이 모두 user하고 database와 상호작용을 한다.

7.3 Subcomponents

7.3.1 Interior

a Class Diagram

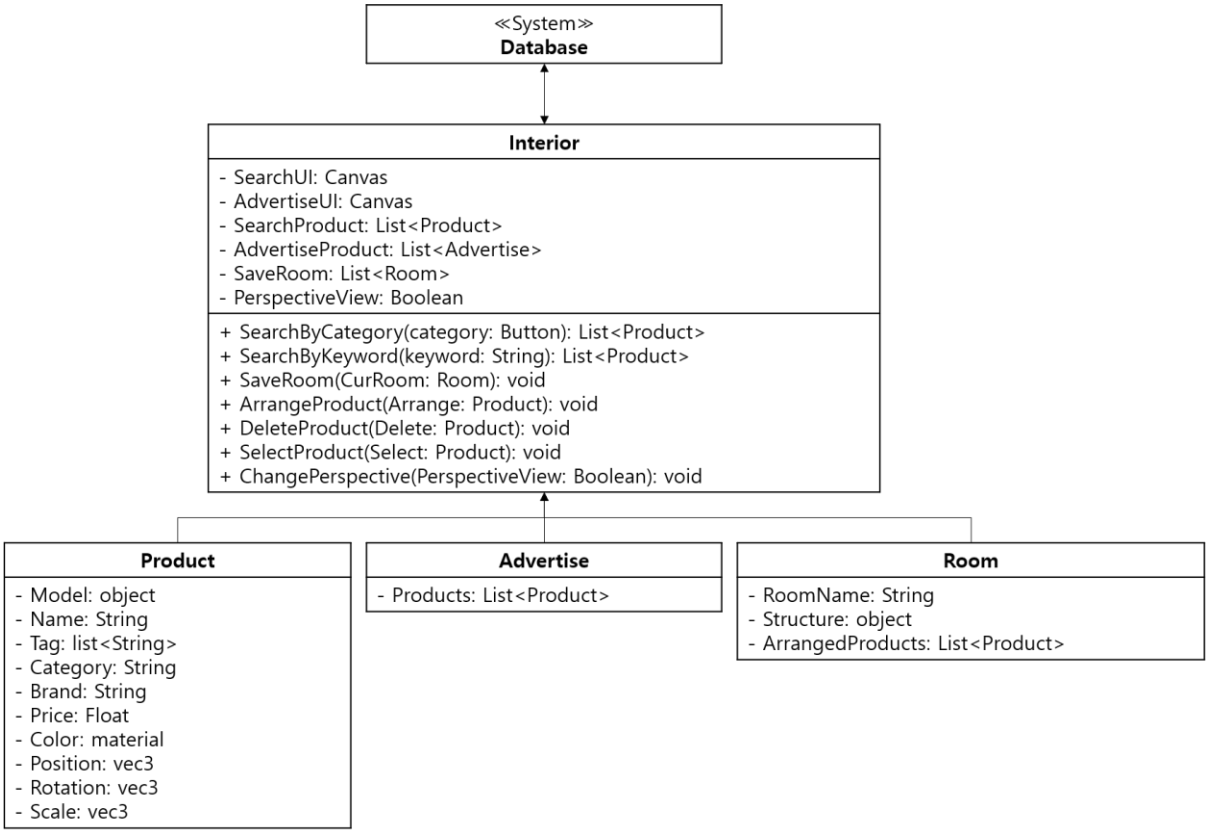


Diagram 24 Interior Class Diagram

i Interior

Attributes

- SearchUI: 사용자가 검색을 하기 위한 UI
- AdvertiseUI: 광고가 사용자에게 노출되기 위한 UI
- SearchProduct: 사용자가 검색한 것에 대응되는 상품 목록
- SaveRoom: 사용자가 저장한 방 목록
- PerspectiveView: 사용자가 현재 보고 있는 관점

Methods

+ SearchByCategory(category: Button): 사용자가 카테고리를 선택하여 검색하기 위한 method
+ SearchByKeyword(keyword: String): 사용자가 키워드를 입력하여 검색하기 위한 method

+ SaveRoom(CurRoom: Room): 사용자가 현재 방을 저장하기 위한 method
+ ArrangeProduct(Arrange: Product): 사용자가 원하는 상품을 방에다가 배치해보기 위한 method
+ DeleteProduct(Delete: Product): 사용자가 배치한 상품을 제외하기 위한 method
+ SelectProduct(Select: Product): 사용자가 상품을 선택하기 위한 method
+ ChangePerspective(PerspectiveView: Boolean): 사용자가 현재 보고 있는 관점을 바꾸기 위한 method

ii Product

Attributes

- Model: 상품의 3D 모델
- Name: 상품의 이름
- Tag: 상품에 해당되는 tag
- Category: 상품이 해당되는 category
- Brand: 상품을 팔고 있는 판매자의 브랜드
- Price: 상품의 가격
- Color: 상품의 색깔 (혹은 option)
- Position: 상품의 위치
- Rotation: 상품의 rotation정보
- Scale: 상품의 크기 정보

iii Advertise

Attributes

- Products: 광고하고 있는 제품 목록

iv Room

Attributes

- RoomName: 현재 방의 이름
- Structure: 현재 방의 구조 (모델)
- ArrangedProducts: 현재 방에 배치 되어있는 상품 목록 (위치 정보 포함)

b Sequence Diagram

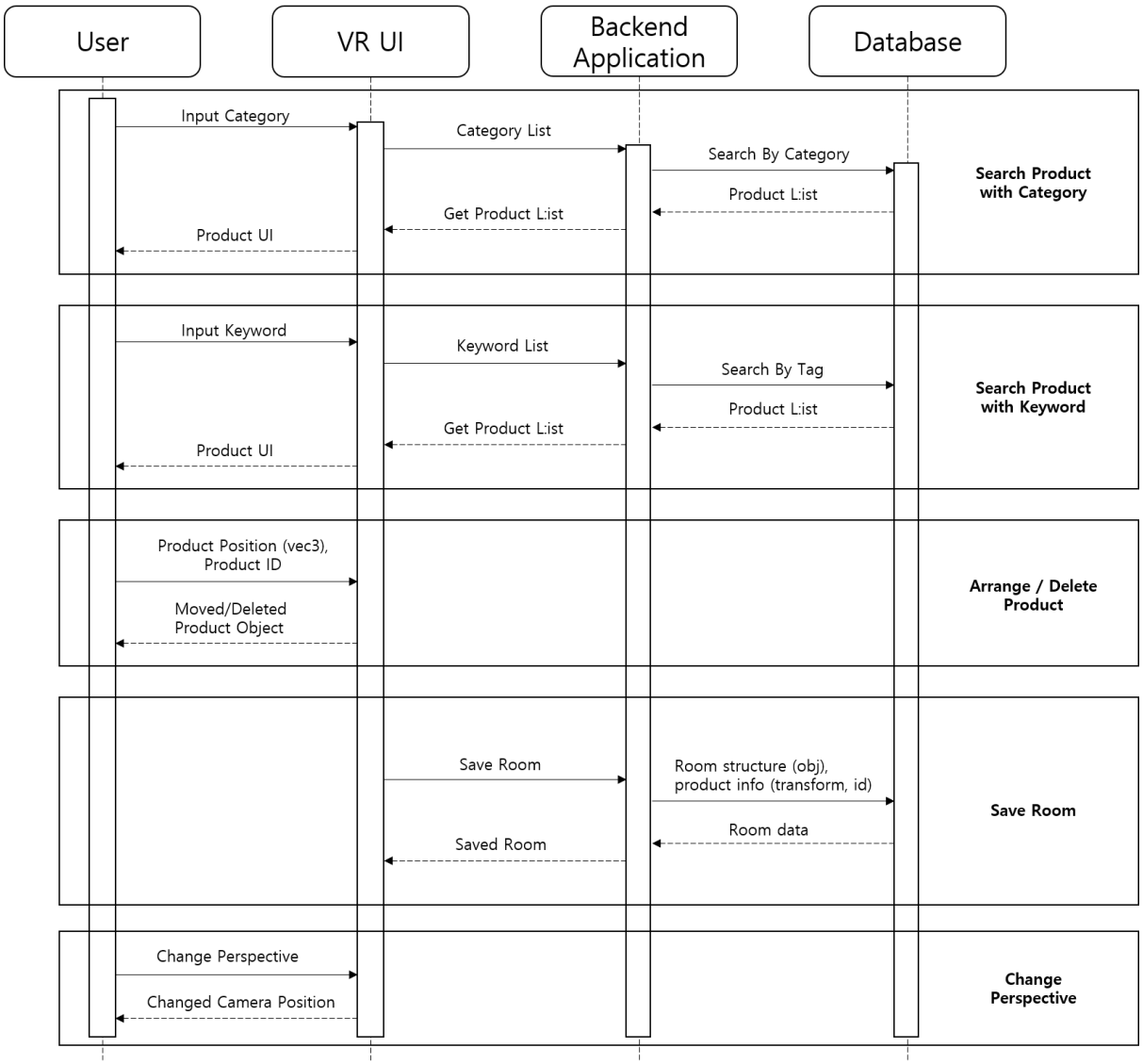


Diagram 25 Interior Sequence Diagram

7.3.2 Cart

a Class Diagram

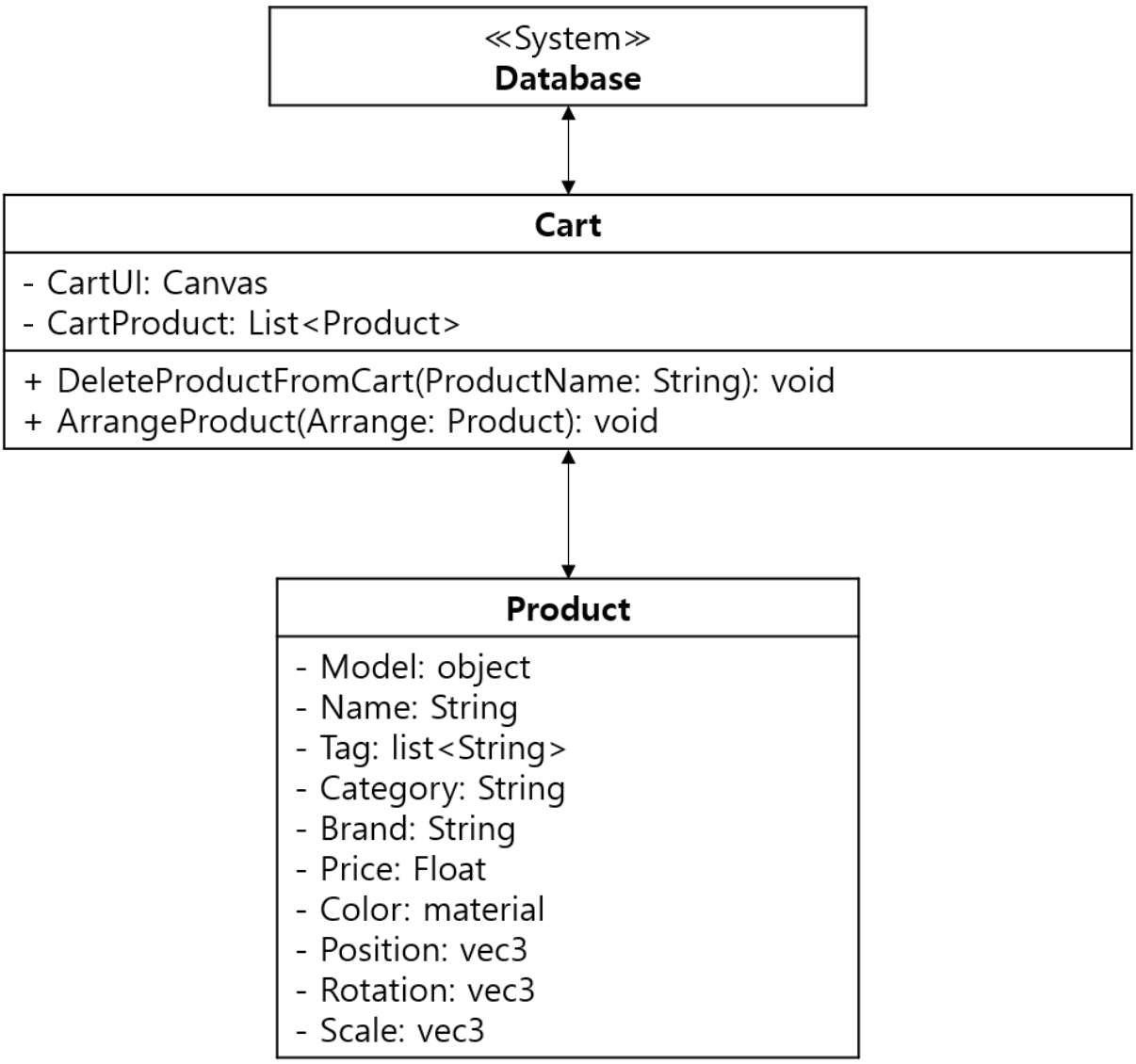


Diagram 26 Cart Class Diagram

i Cart

Attributes

- CartUI: 사용자의 장바구니는 보여줄 UI
- CartProduct: 사용자의 장바구니에 담겨 있는 상품 목록

Methods

+ DeleteProductFromCart(ProductName: String): 장바구니에서 상품을 제외하기 위한 method

+ ArrangeProduct(Arrange: Product): 장바구니에 있는 상품을 방에 배치해보기 위한 method

b Sequence Diagram

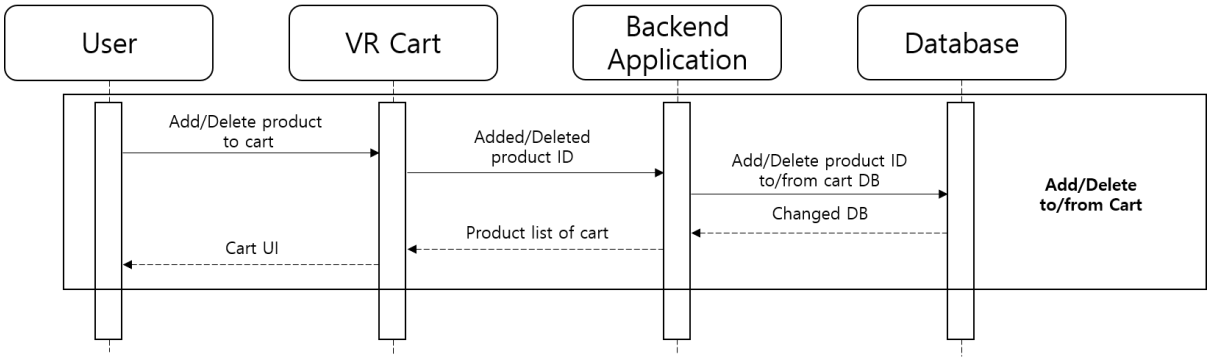


Diagram 27 Cart Sequence Diagram

7.3.3 Purchase

a Class Diagram

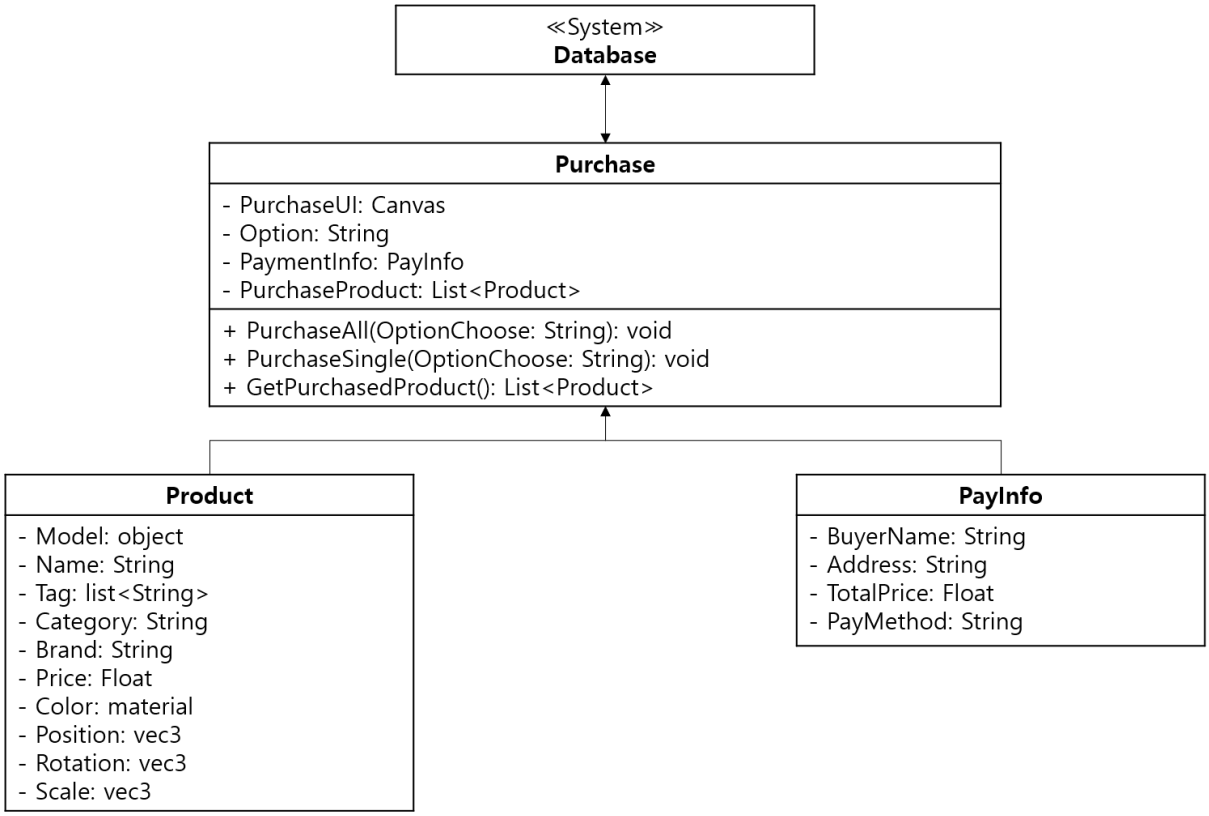


Diagram 28 Purchase Class Diagram

i Purchase

Attributes

- PurchaseUI: 사용자가 구매를 하기 위한 UI
- Option: 사용자가 구매를 할 때 결정한 option 정보
- PaymentInfo: 사용자가 구매를 하기 위해 필요한 정보
- PurchaseProduct: List<Product>: 사용자가 구매를 하는 물품

Methods

+ PurchaseAll(OptionChoose: String): 사용자가 모든 상품을 구매하기 위한 method
+ PurchaseSingle(OptionChoose: String): 사용자가 선택한 상품을 구매하기 위한 method
+ GetPurchasedProduct(): 사용자가 구매한 상품 정보를 보기 위한 method

ii PayInfo

Attributes

- BuyerName: 구매하는 사용자의 이름
- Addresss: 배달 받을 주소
- TotalPrice: 총 가격
- PayMethod: 사용자가 구매를 하기 위해 선택한 방식

b Sequence Diagram

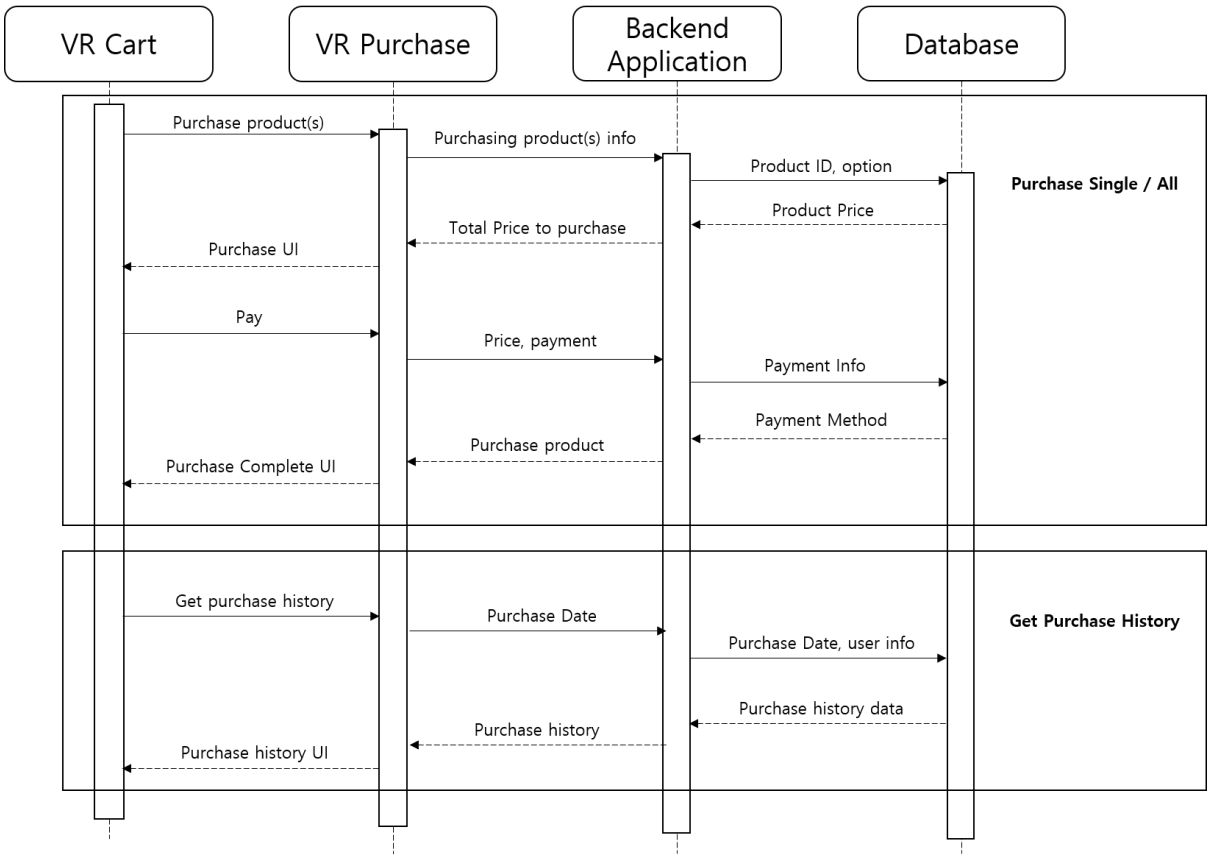


Diagram 29 Purchase Sequence Diagram

8 Database Design

8.1 Objective

Requirement Specification에 기술되어 있는 데이터베이스 요구사항들을 바탕으로 각 Entity를 정의하고 Entity 사이의 relation을 ER Diagram으로 통해 제시한다. 또한 Relation schema를 제공하고, 각 테이블에 대한 DDL 명세를 정의한다.

8.2 ER Diagram

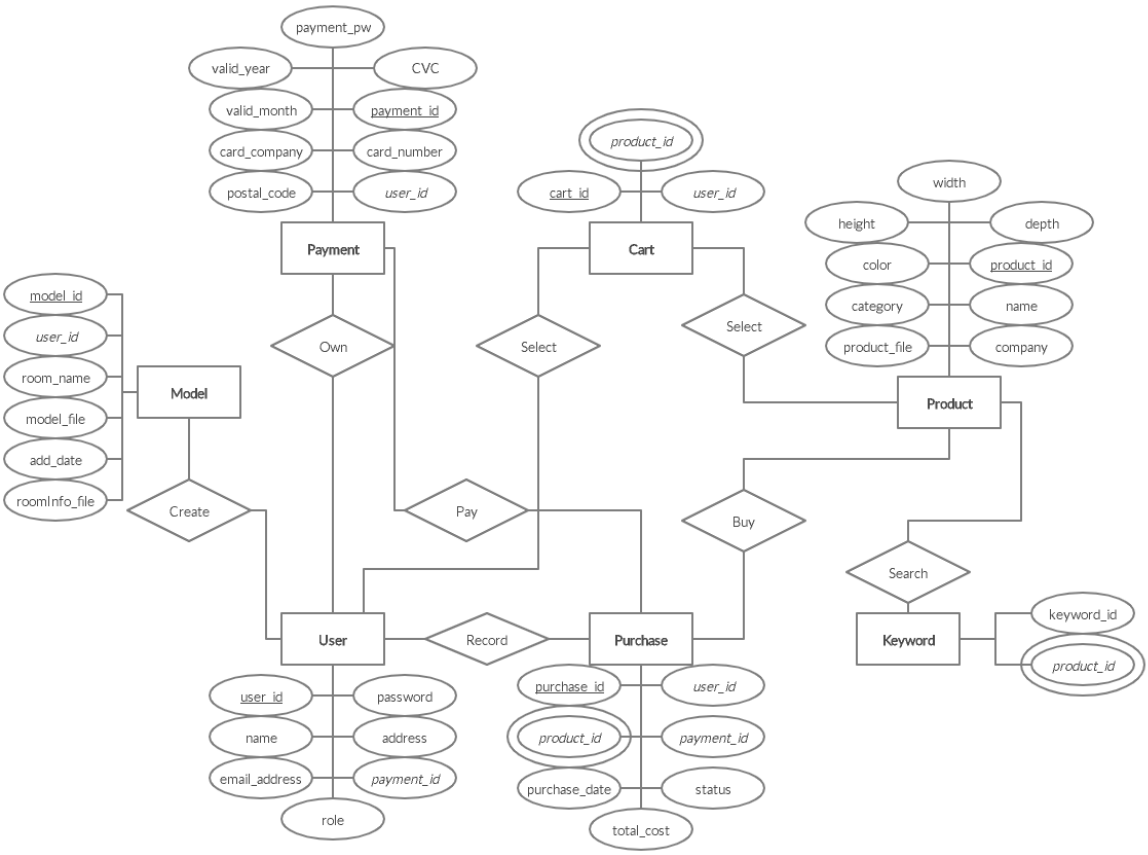


Diagram 30 Database ER Diagram

본 시스템에는 User, Product, Model, Cart, 그리고 Purchase로 구성된 5개의 Entity를 갖고 있으며, 각각의 Entity는 직사각형으로 표현되고, 해당 Entity들 사이의 Relation(관계)는 마름모로 표현된다. 만약 relation에서 다른 Entity와 다대다, 또는 일대다의 관계를 가질 경우 기호를 통해 표현하였다. 각 Entity가 가지는 Attribute는 타원형으로 표현하였으며, 여러 Attribute를 가질 수 있는 경우 이중으로 테두리를 표현하였다. Entity를 식별하는 Primary Key는 밑줄을 그어 표현하였고, 다른 Entity로부터 가져오는 Foreign Key의 경우에는 이탤릭체로 표현하였다.

8.2.1 User

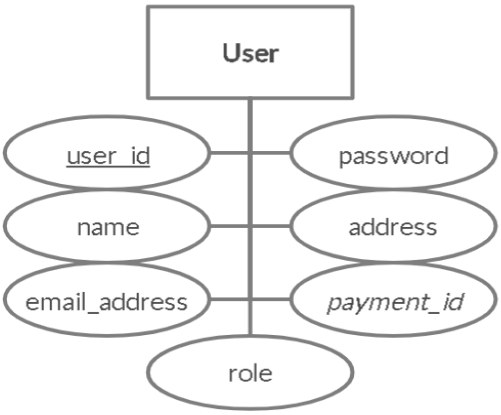


Diagram 31 User ER Diagram

User Entity는 사용자의 정보를 담고 있는 Entity이다. Primary Key는 사용자의 고유 식별번호인 user_id로 제공되며, Attribute에는 비밀번호(password), 이름(name), 주소(address), 이메일 주소(email_address), 결제수단(payment_id), 소비자인지 판매자인지의 여부(role)에 대한 정보를 갖고 있다.

8.2.2 Model

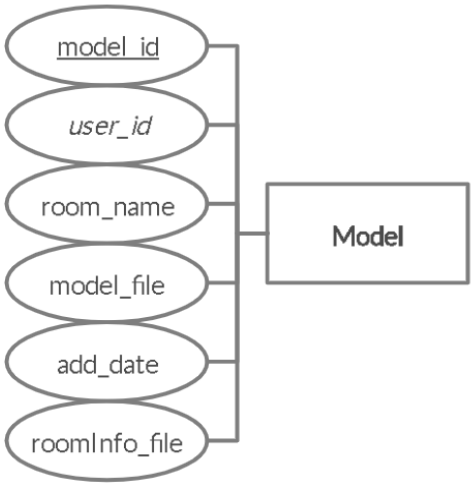


Diagram 32 Model ER Diagram

Model Entity는 사용자가 생성한 방에 대한 모델의 정보를 담고 있는 Entity이다. Primary Key는 생성된 모델의 고유 식별번호인 model_id로 제공되며, Attribute에는 모델을 생성한 사용자의 id(user_id), 생성된 모델의 이름(room_name), 생성된 모델 파일의 이름(model_file), 생성된 날짜(add_date) 그리고 상품의 배치정보를 담고 있는 roomInfo 파일의 파일명(roomInfo_file)으로 구성

된다. 이때 user_id는 Foreign Key로 User Entity에서 가져온 생성한 유저에 대한 id를 갖는다.

8.2.3 Product

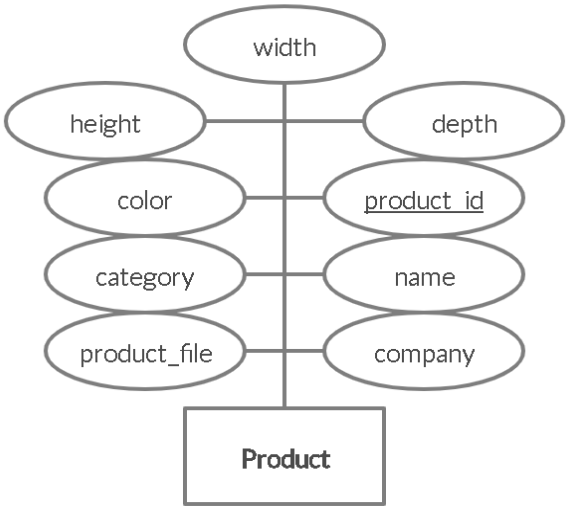


Diagram 33 Product ER Diagram

Product Entity는 가구 또는 상품에 대한 정보를 담고 있는 Entity이다. Primary Key는 상품의 고유 식별번호인 product_id로 제공되며, Attribute에는 상품의 이름(name), 상품의 제조사(company), 상품의 모델 파일의 이름(product_file), 상품의 카테고리(category), 상품의 색상(color), 그리고 상품의 너비, 폭, 높이(width, height, depth)을 담고 있다.

8.2.4 Cart

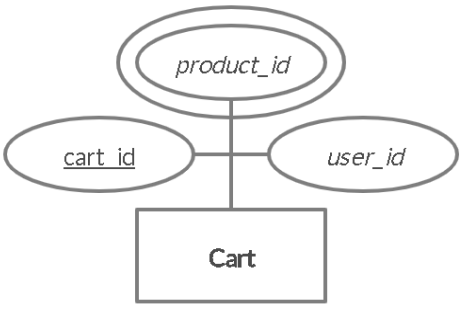


Diagram 34 Cart ER Diagram

Cart Entity는 사용자의 장바구니에 대한 정보를 담고 있는 Entity이다. 사용자는 개인당 하나의 장바구니를 생성할 수 있으며, attribute에는 사용자의 id(user_id)와 장바구니에 담을 상품의 리스트(product_id)를 갖고 있다. product_id와 user_id는 각각 Product Entity와 User Entity에서 가져온

foreign key이다.

8.2.5 Purchase

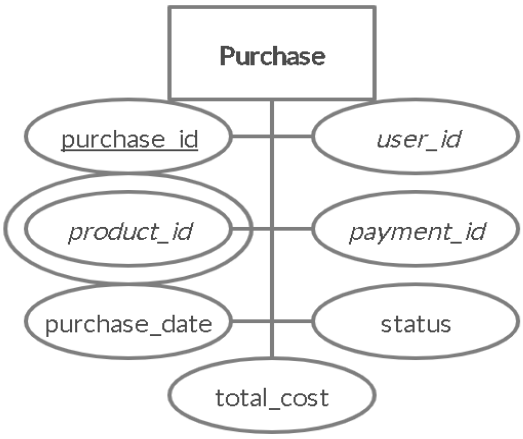


Diagram 35 Purchase ER Diagram

Purchase Entity는 사용자의 구매 내역에 관한 정보를 담고 있는 Entity이다. Primary Key는 각각의 구매 정보 내용을 갖고 있는 purchase_id이다. Attribute에는 구매한 사용자의 id(user_id), 구매한 상품 목록(product_id), 결제정보(payment_id), 구매한 날짜(purchase_date), 반품/환불을 포함한 배송상태(status), 그리고 해당 구매시 총 가격(total_cost)가 있다. product_id attribute에는 한꺼번에 여러 상품을 구매하였을 경우 모든 구매 물품의 리스트를 표현하고, 개별구매를 하였을 경우, 하나의 상품만 나타낸다. user_id, product_id, payment_id는 각각 User Entity, Product Entity, Payment Entity에서 참조하는 foreign key이다.

8.2.6 Payment

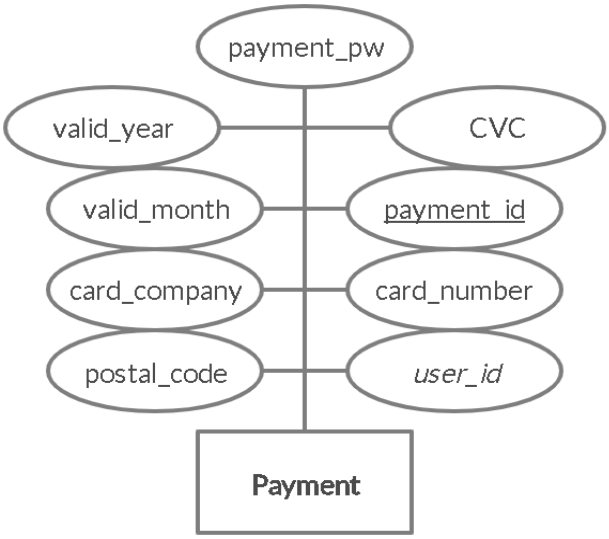


Diagram 36 Payment ER Diagram

Payment Entity는 사용자가 갖고 있는 결제 수단에 대한 정보를 저장하는 Entity이다. 카드 하나 하나의 정보들은 고유의 payment_id를 갖고 있고 이를 Primary Key로 사용한다. 하나의 유저는 여러 개의 결제수단을 소지할 수 있다. Attribute에는 카드를 소유한 사용자의 id(user_id), 배송 코드(postal_code), 카드사(card_company), 카드 번호(card_number), 유효기간 연도/월(valid_year, valid_month), CVC (CVC), 그리고 결제 비밀번호(payment_pw)가 있다. user_id는 User Entity에서 참조하는 foreign key이다.

8.2.7 Keyword

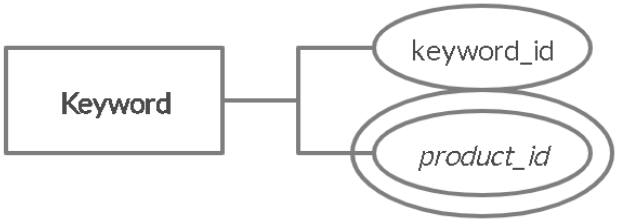


Diagram 37 Keyword ER diagram

Keyword Entity는 상품의 키워드로 분류하기 위한 정보를 담고 있는 Entity이다. 각각의 키워드를 나타내는 keyword_id와 해당 keyword에 포함 되어있는 상품(product_id)을 attribute로 갖고 있다. product_id는 Product Entity에서 참조한 foreign key이다.

8.3 Relational Schema

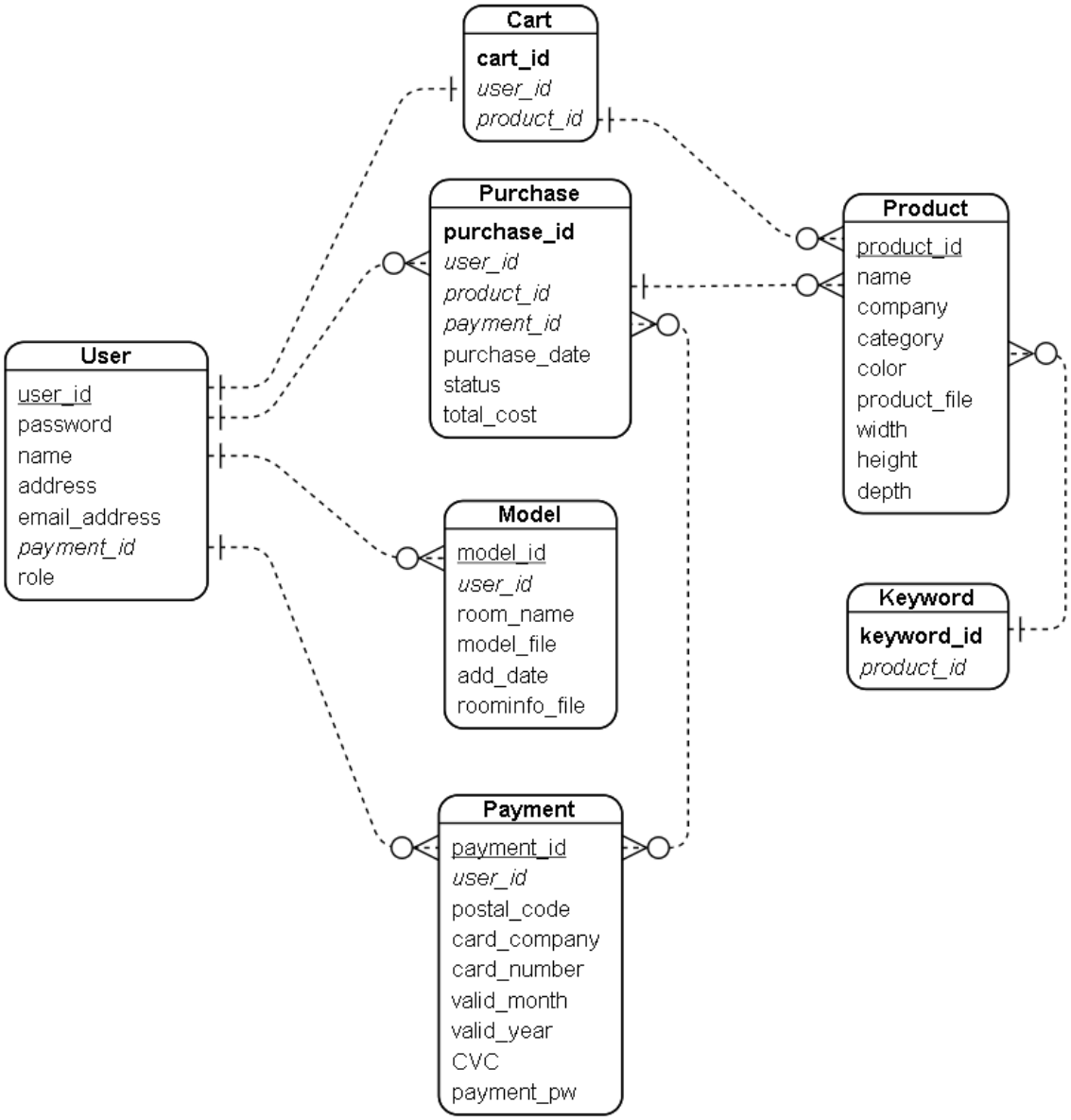


Diagram 38 Relational Schema Diagram

8.4 SQL DDL

8.4.1 User

```
CREATE TABLE USER
(
        user_id VARCHAR(20) NOT NULL,
        password PASSWORD NOT NULL,
        name VARCHAR(30) NOT NULL,
        address VARCHAR(50) NOT NULL,
        email_address VARCHAR(20) NOT NULL,
        payment_id VARCHAR(20) NOT NULL,
        role BOOLEAN NOT NULL,
        PRIMARY KEY (user_id)
);
```

8.4.2 Model

```
CREATE TABLE MODEL
(
        model_id VARCHAR(20) NOT NULL,
        user_id VARCHAR(20) NOT NULL,
        model_file VARCHAR(30) NOT NULL,
        add_date DATE NOT NULL,
        roomInfo_file VARCHAR(30) NOT NULL,
        roomname VARCHAR(20) NOT NULL,
        PRIMARY KEY (model_id),
        FOREIGN KEY (user_id) REFERENCES USER(user_id)
);
```

8.4.3 Product

```
CREATE TABLE PRODUCT
(
        product_id VARCHAR(20) NOT NULL,
        name VARCHARE(20) NOT NULL,
        product_file VARCHAR(30) NOT NULL,
        company VARCHAR(20) NOT NULL,
        width FLOAT NOT NULL,
        height FLOAT NOT NULL,
        depth FLOAT NOT NULL,
        color VARCHAR(20) NOT NULL,
        category VARCHAR(20) NOT NULL,
        PRIMARY KEY (product_id),
);
```

8.4.4 Cart

```
(
        cart_id VARCHAR(20) NOT NULL,
        user_id VARCHAR(20) NOT NULL,
        product_id VARCHAR(20) NOT NULL,
        FOREIGN KEY (user_id) REFERENCES USER(user_id),
        FOREIGN KEY (product_id) REFERENCES PRODUCT(product_id)
);
```

8.4.5 Purchase

```
(
        purchase_id VARCHAR(20) NOT NULL,
        user_id VARCHAR(20) NOT NULL,
        product_id VARCHAR(20) NOT NULL,
        payment_id VARCHAR(20) NOT NULL,
        purchase_date DATE NOT NULL,
        status VARCHAR(10) NOT NULL,
        total_cost INTEGER NOT NULL
        FOREIGN KEY (user_id) REFERENCES USER(user_id),
        FOREIGN KEY (product_id) REFERENCES PRODUCT(product_id),
        FOREIGN KEY (payment_id) REFERENCES PAYMENT(payment_id)
);
```

8.4.6 Payment

```
(
        payment_id VARCHAR(20) NOT NULL,
        user_id VARCHAR(20) NOT NULL,
        postal_code VARCHAR(20) NOT NULL,
        card_company VARCHAR(10) NOT NULL,
        card_number VARCHAR(20) NOT NULL,
        valid_mont INTEGER NOT NULL,
        valid_year INTEGER NOT NULL,
        CVC INTEGER NOT NULL,
        payment_pw PASSWORD NOT NULL,
        PRIMARY KEY (payment_id),
        FOREIGN KEY (user_id) REFERENCES USER(user_id)
);
```

8.4.7 Keyword

```
(
        keyword_id VARCHAR(20) NOT NULL
        product_id VARCHAR(20) NOT NULL,
        FOREIGN KEY (product_id) REFERENCES PRODUCT(product_id)
);
```

9 Testing Plan

9.1 Objective

Component 별로 기능을 구현한 후에 확인 및 점검할 Test 항목을 미리 정의하고 이를 자세하게 기술한다.

9.2 Testing Policy

각 속성별 테스트에 있어 다음의 항목들이 점검되어야 한다.

9.2.1 Backend Application with Database

a Security Testing

- 데이터베이스에 접근하여 데이터를 요청하고자 할 때, 사용자의 권한에 맞는 데이터를 접근할 수 있는가? 사용자가 데이터를 작성, 수정, 삭제하는 과정에서 데이터 베이스의 일관성이 유지되는가?

b Usability Testing

- 사용자가 요청한 정보를 올바르게 처리하고 원하는 데이터를 사용자에게 다시 반환할 수 있는가?

c Accessibility Testing

- 사용자가 정보를 요청하였을 경우 언제든지 요청한 데이터를 제공할 수 있는가?

d Data Access Testing

- 사용자가 정보를 요청한 이후 데이터베이스에서 조건에 맞는 튜플들을 탐색하고 해당 데이터를 반환하는데 걸리는 시간이 얼마나 소요되는가?

9.2.2 Web Application

a Functionality Testing

- 웹 페이지들의 링크들이 제대로 된 목적 페이지로 연결되어 있는가?
- 사용자가 데이터를 작성, 수정, 삭제하는 과정에서 데이터 베이스의 일관성이 유지되는가?

- 데이터를 주고받기 위해 작성하는 폼이 제대로 구성되어 있는가, 잘못된 입력 시 제대로 경고를 생성하는가?
- 쿠키가 정상적으로 생성 및 삭제되는가?

b Usability Testing

- 웹페이지가 사용자가 사용하기 쉬운가?
- 웹페이지에서의 상호작용이 알기 쉽고 명확한가?
- 각 목적을 충족하는 기능들이 알맞게 제공되고 있는가?

c Interface Testing

- 웹 어플리케이션과 데이터베이스 서버의 인터페이스가 정상적으로 작동하는가?
- 인터페이스에서 발생하는 오류들이 적절하게 처리되고 있는가?

d Compatibility Testing

- 다양한 브라우저 환경에서 정상적으로 작동하는가?
- 여러 운영체제 환경에서도 정상적으로 작동하는가?

e Performance Testing

- 한번에 불러와야 할 데이터의 크기가 큰 작업 요청도 잘 처리할 수 있는가(Load Testing)?
- 반복되는 새로고침 등의 부하를 주는 작업에서 정상적으로 작동하는가 문제가 발생한다면 다시 원상 복귀까지 얼마나 시간이 소요되는가(Stress Testing)?

f Security Testing

- 회원 인증 이후 제공되는 페이지로의 비정상적인 접근이 차단되는가?
- 사용자 id, pw와 같은 항목에 잘못된 값을 넣었을 때 더 이상의 진행을 멈추고 정상적으로 경고를 생성하는가?

- 허가되지 않은 웹 경로 혹은 파일로의 접근이 정상적으로 차단되어 있는가?

9.2.3 Room Specify Application

Test Case로 input과 기대하는 결과 output를 미리 준비해두고 input을 입력하여, 동일한 output이출력되는지를 확인한다.

a Functionality Testing

- 미리 ROOM_DESIGN에 쓰일 VERTEX들의 좌표 값, EDGE의 길이, 창문의 크기와 위치를 input으로정해둔다. 기대하는 출력 값을 사전에 계산해 놓는다. Specify Room을 통해서 실제로 output으로 ROOM_DESIGN이 출력되었을 때 이 값과 비교해본다. 이러한 테스트를 모양과 크기가 다른 디자인 5개에 대해서 테스트해본다.
- ROOM_DESIGN을 백엔드를 통해 데이터베이스에 입력해본다. 데이터베이스에 제대로 저장이 되는지 확인한다. 저장된 ROOM_DESIGN을 Specify_Room에서 load해보고, 예상했던 결과와 비교해본다.
- 미리 ROOM_DESIGN에 쓰일 VERTEX들의 좌표, EDGE의 길이, 창문의 크기와 위치를 input으로정해둔다. 기대하는 출력 값을 사전에 계산한다. Specify Room을 통해서 ROOM_DESIGN을 데이터베이스에 저장하고, 데이터베이스에 존재하는 값을 사전에 계산한 결과와 비교해본다. 이러한 테스트를 모양과 크기가 다른 디자인 5개에 대해서 테스트해본다.

b Usability Testing

- 사용자가 사용하기 쉬운가?
- 상호작용이 알기 쉽고 명확한가?
- 각 목적을 충족하는 기능들이 알맞게 제공되고 있는가?

c Interface Testing

- Room Specify 어플리케이션과 데이터베이스 서버의 인터페이스가 정상적으로 작동하는가?
- 인터페이스에서 발생하는 오류들이 적절하게 처리되고 있는가?

d Compatibility Testing

- 다양한 환경에서 정상적으로 작동하는가?

e Performance Testing

- ROOM__DESIGN을 수정할 때에 화면에 빠르게 반영되는가?
- ROOM_DESIGN이 데이터베이스에 입출력 될 때에 빠르게 반영되는가?

9.2.4 Virtual Reality Application

a Functionality Testing

- 사용자가 방에다가 가구를 배치할 때 가구가 방의 구조와 겹치지는 않는가? 혹은 맵을 벗어나지 않는가?
- 사용자의 시야가 실제와 같이 안정적인가?
- 사용자가 물건을 검색하고, 카트에 넣고, 구매를 하는 일련의 과정에 무리는 없는가?
- 오브젝트가 서로 겹치지는 않는가?

b Usability Testing

- 처음 VR을 접한 user도 사용하기에 어렵지 않은가? Teleport와 같은 이동 기술이나 배치 방법, UI input 방법 등의 사용감이 좋은가?
- 사용자가 Vive VR을 사용해서 application을 이용할 수 있는가?

c Performance Testing

- 방 안에 물품들이 50개가 넘어도 문제가 없는가?
- 사용자가 돌아다니는 데에 렉이 걸리지 않는가?

10 Development Plan

10.1 Objective

이번 챕터에서는 실제 개발 단계에서 어떠한 기술과 개발 환경을 사용할 것인지를 기술하고, 개발 일정과 진행 상황을 설명한다.

10.2 Backend Environment + Database

10.2.1 Node.js

Node.js는 확장성 있는 네트워크 애플리케이션(특히 서버 사이드) 개발에 사용되는 소프트웨어 플랫폼이다. 작성 언어로 자바스크립트를 활용하며 Non-blocking I/O와 단일 스레드 이벤트 무한 루프를 통한 높은 처리 성능을 가지고 있다. 내장 HTTP 서버 라이브러리를 포함하고 있어 웹 서버에서 아파치 등 별도의 소프트웨어 없이 동작하는 것이 가능하며 이를 통해 웹서버의 동작에 있어 더 많은 통제를 가능케 한다.

V8 (자바스크립트 엔진)으로 빌드된 이벤트 기반 자바스크립트 런타임으로, 웹 서버와 같이 확장성 있는 네트워크 프로그램 제작을 위해 고안되었다. 서버 측에서 실행되며, 일부 CommonJS 명세를 구현하고 있고, 쌍방향 테스트를 위해 REPL 환경을 포함하고 있다.

10.2.2 MariaDB

MariaDB는 오픈소스의 관계형 데이터베이스 관리 시스템(RDMBS)으로 MySQL과 동일한 소스 코드를 기반으로 하며, GPL v2 라이선스를 따른다. MySQL과 높은 호환성을 유지하며, MySQL API와 명령에 정확히 매칭한다. 사용방법과 구조가 MySQL과 동일하지만. GPL v2 라이선스를 따르는 순수한 오픈소스 프로젝트이기 때문에 오라클로부터 자유롭다.

MariaDB 커뮤니티는 MySQL과 비교해 애플리케이션 부분 속도가 약 4~5천배 정도 빠르며, MySQL이 가지고 있는 모든 제품의 기능을 완벽히 구현하면서도 성능 면에서는 최고 70%의 향상을 보이고 있다고 주장한다.

10.3 Frontend Environment

10.3.1 Vue.js

Vue.js는 발음대로 뷰(View)에 최적화된 프레임워크이다. 컨트롤러 대신 뷰 모델을 가지는 MVVM(Model-View-ViewModel) 패턴을 기반으로 디자인되었으며, 컴포넌트(Components)를 사용하여 재사용이 가능한 UI들을 묶고 뷰 레이어를 정리해주는 강력한 기능을 가졌다. 이러한 싱글 파일 컴포넌트 기법을 통해 필요한 HTML 템플릿, 자바스크립트, CSS 스타일시트 등을 하나의 파일에 모두 작성할 수 있어서 코드 집적도와 유지보수성을 높일 수 있다. 또한 Angular, React와 같은 다른 프론트엔드 프레임워크와 비교했을 때 상대적으로 가볍고 빠르다.

10.4 Room specification Environment

10.4.1 Python

파이썬은 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed) 대화형 언어이다. 다양한 라이브러리를 사용할 수 있는 것이 장점이다. 본 시스템에서는 Specify Room을 기능을 구현하는 데 사용하였다.

10.4.2 OpenCV

OpenCV(Open Source Computer Vision)은 실시간 컴퓨터 비전을 목적으로 한 프로그래밍 라이브러리이다. Specify Room의 기능을 구현할 때 마우스 입출력을 처리해줄 수 있는 모듈이 존재하고, 만든 도형을 시각화 할 수 있다.

10.5 VR Environment

10.5.1 Unity 3D

유니티 3D란, interactive한 컨텐츠를 만들기 위해 사용되는 툴로, 가상환경을 구축하고 설계한다. 실제로 많은 회사와 실무자들이 사용하는 툴이며, 그만큼 방대한 라이브러리와 asset을 가지고 있다. 이 에디터는 윈도우와 맥에서 실행되며, 원하는 형태로 build 파일을 만들 수 있다. 유니티는 그 환경을 보다 더 원하는 데로 설계하기 위해 C# 언어로 스크립트 구성이 가능하도록 했다.

10.5.2 Vive VR, VIVE input utility

Vive VR은 사용자가 inteReal을 이용하기 위해 사용하는 도구로, 가상현실을 보고, interact할 수 있도록 도와준다. 기본적으로 머리에 쓰는 고글 형태의 기기로 가상환경을 볼 수 있으며, 두 개의 컨트롤러로 가상현실의 object와 상호작용이 가능하다. 또한, vive input utility는 unity asset store에서 제공하는 toolkit으로, Vive VR의 input을 컨트롤 하는 것을 도와준다. 이 toolkit은 vive나 vive pro를 목적으로 만들어졌지만, 오큘러스 고, 오큘러스 리프트, 데이드림, 마이크로소프트 mixed reality VR 등 여러 플랫폼에서도 사용 가능하다.

10.6 Schedule

Phase	Expected
Requirements Development	~11/3
Design	~11/11
Backend Development	11/11 ~ 11/20
Web Application Development	11/11 ~ 11/24
SpecifyRoom Development	11/11 ~ 11/24
VR Application Development	11/11 ~ 11/24
Integration	11/24 ~ 11/30
Validation & Verification	11/30 ~ 12/3

Table 1 Schedule Table

11 Index

11.1 Table

11.2 Figures

11.3 Diagram