HW02 : Making the Requirements

1. Project Team

a. 프로젝트 명 : 오크통

b. 팀원:

i. 20213064 이창조 (소프트웨어학부)

군 복무 당시 간부들의 요구사항에 맞춘 군용 웹 프로그래밍 경험과 파이썬 이미지 인식 관련 프로젝트를 한 경험이 있습니다. 다양한 알고리즘 문제를 풀면서 로직 설계 및 구현 경험이 많습니다.

ii. 20212985 남재준 (소프트웨어학부)

파이썬을 통한 데이터 분석과 인공지능을 주제로 간단한 프로젝트 같은 것들을 해본 경험이 있음. 또한 최근에는 c++을 통해 알고리즘을 공부하여 c++에 대한 지식도 갖고 있고, 보안 관련해서도 자격증을 공부하는 과정을 통해 관련 지식도 어느정도 갖고 있다.

2. 추진려는 Project

a. 프로젝트 목표 (vision)

사용자는 본인 취향 위스키의 맛이나 향 등의 데이터를 입력하고 이를 알고리즘이 분석하여 사용자 취향에 딱 맞을 수 있는 다양한 위스키 후보들을 추천하는 것을 목표로 하고 있습니다. 또한 이 과정에서 암호화 및 복호화를 적용하여 사용자 데이터를 보호하는 것도 추가적인 목표로 하고 있습니다.

b. 프로젝트의 범위 (scope)

위스키 취향에 따른 데이터 입력, 위스키 추천받기, 현재 추가한 위스키 목록 확인, 최근 조회한 위스키 목록 확인, 전체 위스키 목록 불러오기, 위스키 목록 필터링 하기, 위스키 목록 정렬하기, 위스키 맛에 대한 평가 노트 작성하기, 선택된 위스키와 유사한 위스키 불러오기, 위스키 상세 정보 보기

3. 기능적인 요구사항 (Functional Requirements)

Use case name	위스키 취향 데이터 입력	
Scenario	사용자가 선호하는 맛들과 가격대를 설정함	
Triggering event	사용자가 위스키 취향 입력탭으로 이동	
Brief description	사용자는 위스키 취향 입력으로 개인 취향을 저장하고 관리할 수 있다. 위스키의 대표적인 4가지 바디감, 풍미, 스모키함, 단맛과 가격적인 정보들에 선호도를 매겨 개인 프로필 정보에 저장한다.	
Actors	사용자	
Related use cases	위스키 추천받기	
Stalkholders	사용자	
Preconditions	사용자의 취향 정보 파일미 미리 DB에 존재해야 한다.	
Post conditions	사용자의 취향 정보가 DB에서 수정된다.	
Flow of activities	Actor	System
	1. 위스키 취향 입력 탭 이동 2. 맛과 가격 선호 정보 입력 3. 저장 버튼 클릭	1.1 위스키 취향 입력 폼을 생성한다. 1.2 맛과 가격 선호정보들을 메모리에 저장한다.

		1.3 메모리에 저장된 정보들을 파일로 저장한다.
Exception condition	1.3 사용자가 모든 입력 항목을 입력하지 않았을 경우 “모든 항목을 입력해주세요"라는 메세지를 화면에 출력한다. 1.3 에러가 발생한 경우 "저장 실패"라는 메세지를 출력하여 사용자의 취향 입력 기능을 다시 사용할 것을 유도한다.	

Use case name	위스키 추천받기	
Scenario	사용자가 입력한 취향을 기반으로 위스키를 추천함	
Triggering event	사용자가 위스키 추천 탭으로 이동	
Brief description	사용자는 추천 기능을 이용하여 사전에 입력된 취향 정보를 바탕으로 취향에 가장 적절한 위스키 후보들을 볼 수 있다.	
Actors	사용자	
Related use cases	위스키 취향 데이터 입력	
Stalkholders	사용자, 서비스 제공자	
Preconditions	사용자의 취향 정보에 미리 정보들이 입력되어 있어야 한다.	
Post conditions	사용자에게 추천 위스키 목록이 표시됨	
Flow of activities	Actor	System
	1. '위스키 추천' 메뉴 선택	1.1 사용자 프로필에서 취향 데이터 로드 1.2 추천 알고리즘 실행 1.3 추천 위스키 목록 생성

		1.4 사용자에게 추천 위스키 목록 표시
Exception condition	2.2 추천 알고리즘 실행 중 오류 발생 시, 시스템은 "추천 시스템 오류가 발생했습니다." 메시지를 표시하여 추천 기능 재시도를 유도한다.	

Use case name	추가한 위스키 목록 확인	
Scenario	사용자가 직접 평가하고 저장한 위스키 목록을 확인	
Triggering event	사용자가 '내가 평가한 위스키' 메뉴를 선택함	
Brief description	사용자는 자신이 평가한 위스키들을 확인하고 평가들을 추가하거나 수정할 수 있다.	
Actors	사용자	
Related use cases	위스키 취향 데이터 입력	
Stalkholders	사용자	
Preconditions	사용자 정보가 존재해야 함	
Post conditions	사용자가 평가한 위스키 목록과 평가 점수가 화면에 표시됨	
Flow of activities	Actor	System
	1. '내가 평가한 위스키' 메뉴 선택 2. 특정 위스키 선택 3 위스키에 대한 평가 추가	1.1 사용자 프로필 DB에서 평가한 위스키 목록 및 점수 조회 2.1 해당 위스키에 대한 수정 버튼이 생성됨

		3.1 위스키에 대한 평가를 내릴 수 있는 폼으로 이동
Exception condition	1.1 해당 위스키 평에 대한 정보를 찾을 수 없는 경우 “위스키 평에 대한 정보를 찾을 수 없습니다"라는 메세지를 출력한다. 3.1 위스키 평가 저장에서 에러가 발생한 경우 “저장중 오류가 발생하였습니다"라는 메세지를 사용자 화면에 출력한다	

Use case name	최근 조회한 위스키 목록 확인	
Scenario	사용자가 최근에 조회했던 위스키 목록을 확인	
Triggering event	사용자가 '최근 본 위스키' 메뉴를 선택함	
Brief description	사용자는 최근에 상세정보를 확인했던 위스키들을 확인할 수 있다. 위스키 정보들을 조회 시간을 기준으로 오름차순으로 화면에 출력한다.	
Actors	사용자	
Related use cases	없음	
Stalkholders	사용자	
Preconditions	사용자 정보가 존재해야 함	
Post conditions	최근 조회한 위스키 목록이 시간 순서대로 화면에 표시됨	
Flow of activities	Actor	System
	1. '최근 본 위스키' 메뉴 선택	1.1 사용자의 위스키 조회 기록 데이터 불러오기

		1.2 조회 기록을 시간 순서대로 정렬 1.3 정렬된 목록을 화면에 표시
Exception condition	1.1 최근 조회한 위스키 목록이 비었을 경우 “최근 조회한 위스키가 없습니다"라는 메세지를 출력한다. 1.2 조회와 정렬에서 오류가 발생한 경우 “조회 기록 로딩 중 오류가 발생하였습니다"라는 메세지를 사용자 화면에 출력한다.	

Use case name	전체 위스키 목록 불러오기	
Scenario	데이터베이스에 있는 모든 위스키 목록을 확인	
Triggering event	사용자가 '전체 위스키' 또는 '위스키 검색' 메뉴 등을 통해 전체 목록 조회를 시도함	
Brief description	위스키 전체에 대한 목록을 확인하고 세부정보 확인이나 검색, 정렬과 같은 부가 기능에 대해 접근할 수 있다.	
Actors	사용자	
Related use cases	없음	
Stalkholders	사용자, 서비스 제공자	
Preconditions	사전에 전체 위스키 목록들이 DB에 저장되어 있어야 함	
Post conditions	전체 위스키 목록이 화면에 표시됨	
Flow of activities	Actor	System
	1. '전체 위스키' 메뉴 선택 2. 스크롤 또는 '다음 페이지' 클릭	1.1 DB에 전체 위스키 목록 조회 요청

		1.2 DB에서 위스키 목록 검색 및 반환 1.3 화면에 첫 페이지 위스키 목록 표시 2.1 요청 시에 다음 페이지 데이터 로드 및 표시
Exception condition	2.1 위스키 전체 목록 파일이 누락되거나 찾을 수 없는 경우 “위스키 데이터를 불러올 수 없습니다"라는 메세지를 사용자 화면에 출력한다. 2.2 전체 위스키 목록에 너무 오랜시간이 걸리는 경우 일부 데이터만 사용자 화면에 출력한 후 다음 페이지 보기 버튼을 활성화 시킨다.	

Use case name	위스키 목록 필터링 하기
Scenario	위스키 목록을 맛, 생산지, 종류 등으로 필터링하여 원하는 위스키 목록만 조회
Triggering event	사용자가 전체 위스키 목록 기능을 이용하던 중 필터링 버튼을 눌러 필터링 조건을 활성화 시킴
Brief description	가격과 같은 위스키 정보들을 사용자가 설정한 필터링 조건을 기반으로 걸러내어 화면에 출력함
Actors	사용자
Related use cases	전체 위스키 목록 불러오기, 위스키 정보 상제 적용
Stalkholders	사용자, 서비스 제공자
Preconditions	시스템에 위스키 목록이 등록되어 있어야 한다.

Post conditions	필터링된 위스키 목록들을 불러온 후 사용자 화면에 출력한다.	
Flow of activities	Actor	System
	1.종류나 가격 등 필터 조건을 설정한다 2.”필터 적용”을 요청한다	1.1 입력된 필터 조건을 검증한다. 1.2 DB에서 조건에 해당하는 위스키 목록을 조회한다. 2.1 조회된 위스키 목록을 화면에 표시한다.
Exception condition	2.1 해당 필터링에 검출된 위스키를 찾을 수 없는 경우 “위스키를 찾을 수 없습니다”라는 메세지를 사용자 화면에 출력한다.	

Use case name	위스키 목록 정렬하기
Scenario	사용자가 위스키 목록을 가격, 평점, 숙성 연도 등 다양한 정렬 기준으로 정렬하여 원하는 순서대로 조회
Triggering event	사용자가 위스키 목록 화면에 진입하여 특정 정렬 기준을 선택하거나 “정렬 적용” 버튼을 클릭한다.
Brief description	사용자가 현재 보고 있는 위스키 목록들을 조건을 설정하여 해당 조건을 기준으로 정렬하여 다시 볼 수 있게한다.
Actors	사용자
Related use cases	전체 위스키 목록 불러오기, 위스키 정보 상세 조회
Stalkholders	사용자, 서비스 제공자
Preconditions	시스템에 위스키 목록과 해당 위스키들의 세부정보가 미리 등록되어 있어야 한다.

Post conditions	설정된 정렬 조건들을 기반으로 위스키 목록들이 재생성되며 출력된다.	
Flow of activities	Actor	System
	1.정렬 기준을 선택한다. 2.”정렬 적용” 버튼을 클릭한다. 3.필요 시 다른 정렬 기준으로 변경하여 재정렬을 요청한다.	1.1정렬기준이 유효한 조건인지 검증한다. 1.2 위스키 목록들을 정렬기준을 기반으로 정렬된다. 2.1 정렬된 위스키 목록을 화면에 표시한다.
Exception condition	2.1 해당 정렬 조건으로 정렬을 수행하다 오류가 발생한 경우 “정렬 도중 오류가 발생하였습니다. 다시 시도하여 주세요.” 라는 메세지를 출력한다. 2.2 정렬 결과가 없을 경우 “해당 조건의 위스키를 찾을 수 없습니다"라는 메세지를 출력한다.	

Use case name	위스키 맛에 대한 평가 노트 작성하기
Scenario	사용자가 특정 위스키에 대해 평가노트를 작성하여 저장함
Triggering event	사용자가 위스키 상세 정보에서 “평가 노트 작성" 버튼을 클릭
Brief description	사용자가 실제로 위스키를 시음해본 후 맛이나 풍미등 평가 항목을 코멘트와 함께 평가노트를 작성하여 저장함
Actors	사용자
Related use cases	위스키 상세 정보 조회, 사용자 리뷰 관리
Stalkholders	사용자, 서비스 제공자

Preconditions	시스템에 평가 대상 위스키의 상세 정보가 등록되어 있어야 한다.	
Post conditions	사용자가 작성한 평가 노트가 시스템에 저장되어 위스키 상세 페이지에 업데이트된다.	
Flow of activities	Actor	System
	1.사용자가 위스키 상세 페이지에서 “평가 노트 작성” 버튼을 클릭한다. 2. 사용자가 해당 위스키에 대해 각종 평가를 내린 평가노트를 작성한다. 3.사용자가 “저장” 버튼을 클릭한다.	1.1 평가 노트 작성 화면을 표시한다. 2.1 평가 노트를 데이터베이스에 저장한다. 3.1 저장 완료 후, 평가 노트를 위스키 상세 페이지에 반영하여 표시한다.
Exception condition	3.1 평가 노트에 저장 과정중 오류가 발생한 경우 “평가 노트를 작성할 수 없습니다. 오류가 발생하였습니다"라는 메세지를 화면에 출력한다.	

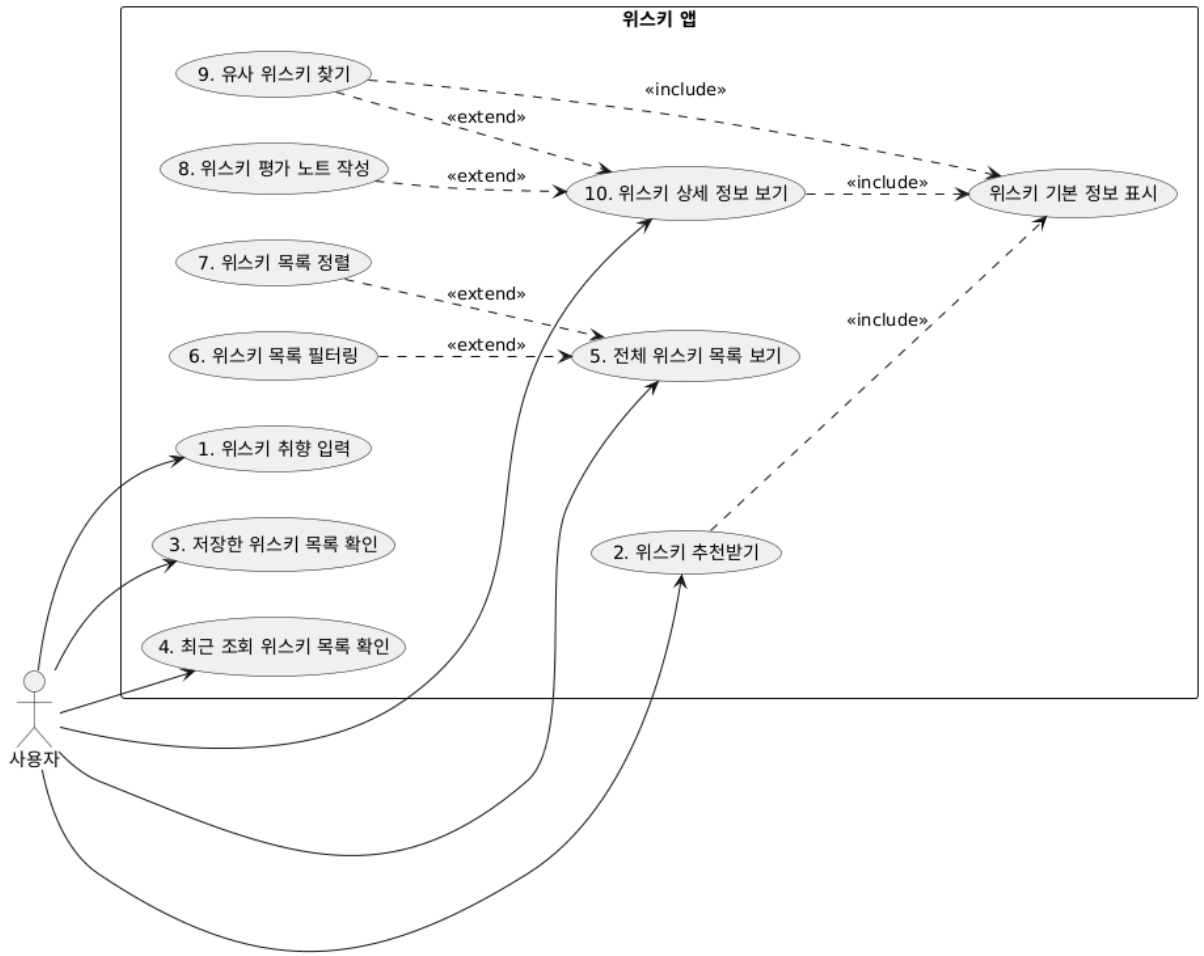
Use case name	선택된 위스키와 유사한 위스키 불러오기
Scenario	사용자가 특정 위스키를 선택한 후, 해당 위스키와 맛, 가격, 평가 등 다양한 특성을 비교하여 유사한 위스키를 추천받는다.
Triggering event	사용자가 위스키 상세 정보 페이지에서 “유사한 위스키 보기” 또는 “추천” 버튼을 클릭함.
Brief description	해당 위스키가 마음에 들어 관련된 위스키들을 더 알아보고 싶은 경우 해당 위스키의 세부정보들을 기반으로 알고리즘을 작동시켜 비슷한 위스키들을 알아볼 수 있다. 만약 결과가 없을 경우 그 과정을 출력하여 사용자에게 알린다.

Actors	사용자	
Related use cases	위스키 상세 정보 조회 위스키 추천 알고리즘 실행	
Stalkholders	사용자, 서비스 제공자	
Preconditions	위스키들의 상세정보가 미리 저장이 되어 있어서 이를 기반으로 하는 위스키간 유사성 계산 알고리즘이 적용되어 있어야 한다.	
Post conditions	기능을 적용하면 선택된 위스키와 유사한 위스키들이 화면에 표시되어 사용자에게 보여진다. 추천 결과를 찾을 수 없거나 로직상 문제가 발생했을 때에는 적절한 안내가 메세지 형태로 표시되며 오류 발생시 로그가 저장되어 추후 개선에 사용된다.	
Flow of activities	Actor	System
	1.사용자가 위스키 상세 정보 페이지에서 “유사한 위스키 보기” 버튼을 클릭한다. 2. 추천을 진행한다. 3.추천 결과를 확인한다.	1.1 선택된 위스키의 맛, 가격, 평가 등의 특성을 확인한다. 2.1 유사성 계산 알고리즘을 실행하여 해당 위스키와 유사한 위스키를 선별한다. 3.1 계산결과 유사한 위스키로 판정된 위스키들을 화면에 표시한다.
Exception condition	2.1 알고리즘 실행 과정에서 오류가 발생한 경우 시스템이 “추천 정보 생성 중 오류 발생"이라는 메세지를 사용자에게 표시한다. 2.2 알고리즘상 유사한 위스키를 선별할 수 없는 경우 시스템은 “유사한 위스키를 찾을 수 없습니다"라는 메세지를 사용자에게 안내한다.	

Use case name	위스키 상세 정보 보기	
Scenario	사용자가 위스키 목록에서 위스키를 선택하면 해당 위스키의 상제한 정보들을 확인할 수 있다.	
Triggering event	사용자가 위스키 목록에서 특정 위스키를 클릭한다.	
Brief description	사용자가 목록에서 특정 위스키를 클릭하면 시스템은 해당 위스키에 대한 상세정보를 DB에서 검색한 후 사용자 화면에 출력한다. 정보를 찾을 수 없는 경우 “정보 없음"이라는 메세지를 출력한다.	
Actors	사용자	
Related use cases	전체 위스키 목록 불러오기, 위스키 평가 노트 작성하기, 위스키 정보 업데이트	
Stalkholders	사용자, 서비스 제공자	
Preconditions	DB에 위스키에 대한 상세정보가 미리 등록되어 있어야 한다.	
Post conditions	사용자가 선택한 특정 위스키의 상세정보가 화면에 출력된다. 찾을 수 없는 위스키 정보의 경우 “정보 없음"이라는 대체 메세지가 출력된다.	
Flow of activities	Actor	System
	1.사용자가 위스키 목록에서 특정 위스키를 확인한다. 2. 특정 위스키를 클릭한다.	1.1선택된 위스키의 식별 정보를 확인한다. 2.1데이터베이스에서 해당 위스키의 상세 정보를 조회한다. 2.2 찾은 상세정보를 사용자 화면에 출력한다.

Exception condition	2.1 선택된 특정 위스키에 대한 정보를 찾을 수 없는 경우 “정보 없음" 이라는 메세지를 사용자 화면에 출력한다. 2.2 조회된 상세정보 중 누락된 정보가 있을 경우 해당 부분만 “정보 없음" 메세지로 대체되어 출력한다.

3-2. Use case Diagram



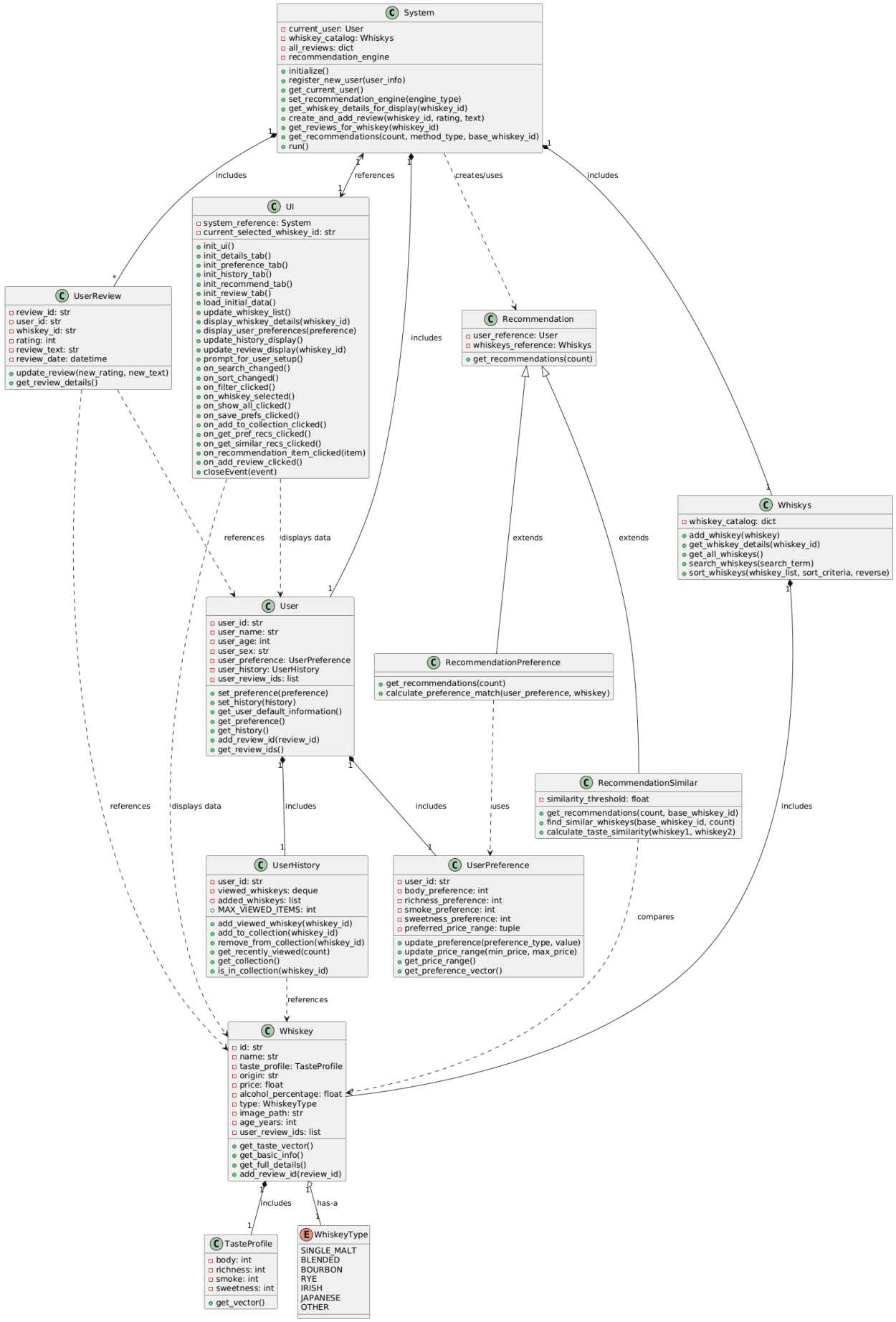
4. 비기능적인 요구 사항 (Non-Functional Requirements)

해당 Use Case / 업무	Non-Functional Requirement 내용	Quality	Quality Attributes

위스키 취향 데이터 입력하기	사용자가 취향 정보를 입력하면 해당 내용이 신속하게 DB에 적용되어야 한다.	Performance Efficiency	평균 2초 이내에 DB에 적용된다.
위스키 추천받기	사용자가 취향에 맞는 위스키 추천받기를 사용할때에는 결과가 빠르게 나와야 한다.	Performance Efficiency	평균 3초 이내에 결과가 화면에 출력된다.
추가한 위스키 목록 확인하기	추가 위스키의 목록을 확인할 때 신속하게 결과가 나와야 한다.	Performance Efficiency	평균 2초 이내에 추가한 위스키들을 출력한다.
최근 조회한 위스키 목록 확인하기	사용자가 최근 조회한 위스키들을 안정적으로 DB에서 관리되어야 하며 조회시에도 신속하게 결과를 도출한다.	Performance Efficiency, Reliability	평균 2초 이내에 최근 조회 위스키들을 불러오며 파일 저장 오류율을 5% 이내로 한다.
전체 위스키 목록 불러오기	사용자가 전체 위스키 목록을 불러 올때 가능한한 모든 위스키들을 불러오며 신속해야 한다.	Reliability	Availability 를 98%이상으로 유지하며 평균 5초 이내에 위스키 목록들을 불러온다.
위스키 목록 필터링하기	사용자가 필터링 조건을 설정했을때 신속하게 위스키 목록들에 반영되어야 한다.	Performance Efficiency	평균 2초 이내에 필터링 결과를 화면에 출력한다.

위스키 목록 정렬하기	사용자가 위스키 목록을 정렬하는 요청을 했을 때 신속하게 결과를 도출해야 한다.	Performance Efficiency	평균 2초 이내에 정렬된 위스키 목록을 출력한다.
위스키 맛에 대한 평가 노트 작성하기	작성한 평가 노트는 신속하게 저장되며 불러와져야 한다.	Performance Efficiency	평균 1초 이내에 평가 노트를 DB에 젖아하며 평균 2초 이내에 선택된 평가 노트를 불러온다.
선택된 위스키와 유사한 위스키 불러오기	선택한 위스키와 유사한 위스키들은 신속하게 불러와져야 한다	Performance Efficiency	평균 3초 이내에 선택된 위스키들과 유사한 위스키들을 화면에 출력한다.
위스키 상세 정보 보기	사용자가 요청한 위스키의 상세정보는 신속하게 표시되어야 하며 누락된 정보들도 직관적으로 표시되어야 한다.	Performance Efficiency, Usability	평균 1초 이내에 해당 정보들을 출력하며 누락된 정보들을 “정보 없음"으로 표시한다.

5. Conceptual Class


System
- current_user: User
- whiskey_catalog: Whiskys
- all_reviews: dict
- recommendation_engine
+ initialize()
+ register_new_user(user_info)
+ get_current_user()
+ set_recommendation_engine(engine_type)
+ get_whiskey_details_for_display(whiskey_id)
+ create_and_add_review(whiskey_id, rating, text)
+ get_reviews_for_whiskey(whiskey_id)
+ get_recommendations(count, method_type, base_whiskey_id)
+ run()
UI
- system_reference: System
- current_selected_whiskey_id: str
+ init_ui()
+ init_details_tab()
+ init_preference_tab()
+ init_history_tab()
+ init_recommend_tab()
+ init_review_tab()
+ load_initial_data()
+ update_whiskey_list()
+ display_whiskey_details(whiskey_id)
+ display_user_preferences(preference)
+ update_history_display()
+ update_review_display(whiskey_id)
+ prompt_for_user_setup()
+ on_search_changed()
+ on_sort_changed()
+ on_filter_clicked()
+ on_whiskey_selected()
+ on_show_all_clicked()
+ on_save_prefs_clicked()
+ on_add_to_collection_clicked()
+ on_get_pref_recs_clicked()
+ on_get_similar_recs_clicked()
+ on_recommendation_item_clicked(item)
+ on_add_review_clicked()
+ closeEvent(event)
UserReview
- review_id: str
- user_id: str
- whiskey_id: str
- rating: int
- review_text: str
- review_date: datetime
+ update_review(new_rating, new_text)
+ get_review_details()
Recommendation
- user_reference: User
- whiskeys_reference: Whiskys
+ get_recommendations(count)
Whiskys
- whiskey_catalog: dict
+ add_whiskey(whiskey)
+ get_whiskey_details(whiskey_id)
+ get_all_whiskeys()
+ search_whiskeys(search_term)
+ sort_whiskeys(whiskey_list, sort_criteria, reverse)
User
- user_id: str
- user_name: str
- user_age: int
- user_sex: str
- user_preference: UserPreference
- user_history: UserHistory
- user_review_ids: list
+ set_preference(preference)
+ set_history(history)
+ get_user_default_information()
+ get_preference()
+ get_history()
+ add_review_id(review_id)
+ get_review_ids()
RecommendationPreference
+ get_recommendations(count)
+ calculate_preference_match(user_preference, whiskey)
RecommendationSimilar
- similarity_threshold: float
+ get_recommendations(count, base_whiskey_id)
+ find_similar_whiskeys(base_whiskey_id, count)
+ calculate_taste_similarity(whiskey1, whiskey2)
UserHistory
- user_id: str
- viewed_whiskeys: deque
- added_whiskeys: list
- MAX_VIEWED_ITEMS: int
+ add_viewed_whiskey(whiskey_id)
+ add_to_collection(whiskey_id)
+ remove_from_collection(whiskey_id)
+ get_recently_viewed(count)
+ get_collection()
+ is_in_collection(whiskey_id)
UserPreference
- user_id: str
- body_preference: int
- richness_preference: int
- smoke_preference: int
- sweetness_preference: int
- preferred_price_range: tuple
+ update_preference(preference_type, value)
+ update_price_range(min_price, max_price)
+ get_price_range()
+ get_preference_vector()
Whiskey
- id: str
- name: str
- taste_profile: TasteProfile
- origin: str
- price: float
- alcohol_percentage: float
- type: WhiskeyType
- image_path: str
- age_years: int
- user_review_ids: list
+ get_taste_vector()
+ get_basic_info()
+ get_full_details()
+ add_review_id(review_id)
TasteProfile
- body: int
- richness: int
- smoke: int
- sweetness: int
+ get_vector()
WhiskeyType
SINGLE_MALT
BLENDED
BOURBON
RYE
IRISH
JAPANESE
OTHER
includes
references
creates/uses
includes
includes
extends
extends
references
displays data
includes
includes
uses
includes
compares
references
displays data
includes
includes
has-a

user.py

```python
class User:
    def __init__(self, user_id, user_name, user_age=None, user_sex=None):
        self.user_id = user_id
        self.user_name = user_name
        self.user_age = user_age
        self.user_sex = user_sex

        # 관련 객체들
        self.user_preference = None
        self.user_history = None
        self.user_review_ids = []

    def set_preference(self, preference):       You, 7일 전 • 코드 압축
        pass

    def set_history(self, history):
        pass

    def get_user_default_information(self):
        pass

    def get_preference(self):
        pass

    def get_history(self):
        pass

    def add_review_id(self, review_id):
        pass

    def get_review_ids(self):
        pass
```

whiskey.py

```python
from taste_profile import TasteProfile
You, 6일 전 | 1 author (You)
class Whiskey:
    def __init__(self, whiskey_id, name, taste_profile, origin, price,
                 alcohol_percentage, whiskey_type, image_path=None, age_years=None):
        self.id = whiskey_id
        self.name = name
        self.taste_profile = taste_profile
        self.origin = origin
        self.price = price
        self.alcohol_percentage = alcohol_percentage
        self.type = whiskey_type
        self.image_path = image_path
        self.age_years = age_years
        self.user_review_ids = []

    def get_taste_vector(self):
        pass

    def get_basic_info(self):
        pass

    def get_full_details(self):
        pass

    def add_review_id(self, review_id):      You, 7일 전 • 코드 압축
        pass

    def __str__(self):
        pass
```

whiskeys.py

```python
from whiskey import Whiskey

You, 6일 전 | 1 author (You)
class Whiskys:
    def __init__(self):
        self.whiskey_catalog = {}  # 위스키 ID -> Whiskey 객체

    def add_whiskey(self, whiskey):
        pass

    def get_whiskey_details(self, whiskey_id):
        pass

    def get_all_whiskeys(self):
        pass

    def search_whiskeys(self, search_term):
        pass

    def sort_whiskeys(self, whiskey_list, sort_criteria, reverse=False):
        pass      You, 6일 전 • 뼈대 구성
```

user_review.py

```python
import datetime

You, 6일 전 | 1 author (You)
class UserReview:
    def __init__(self, review_id, user_id, whiskey_id, rating, review_text=""):
        self.review_id = review_id
        self.user_id = user_id
        self.whiskey_id = whiskey_id
        self. (variable) review_text: str
        self.review_text = review_text
        self.review_date = datetime.datetime.now()

    def update_review(self, new_rating=None, new_text=None):      You, 7일 전 • 코드 압축
        pass

    def get_review_details(self):
        pass

    def __str__(self):
        pass
```

user_preference.py

```python
class UserPreference:      You, 6일 전 • 뼈대 완성 …
    def __init__(self, user_id):
        self.user_id = user_id
        self.body_preference = 3
        self.richness_preference = 3
        self.smoke_preference = 3
        self.sweetness_preference = 3
        self.preferred_price_range = (None, None)  # (최소, 최대)

    def update_preference(self, preference_type, value):
        pass

    def update_price_range(self, min_price, max_price):
        pass

    def get_price_range(self):
        pass

    def get_preference_vector(self):
        pass
```

user_history.py

```python
import datetime
from collections import deque

You, 6일 전 | 1 author (You)
class UserHistory:
    MAX_VIEWED_ITEMS = 20

    def __init__(self, user_id):
        self.user_id = user_id
        self.viewed_whiskeys = deque(maxlen=self.MAX_VIEWED_ITEMS)  # (위스키ID, 조회시각)
        self.added_whiskeys = []  # 컬렉션에 추가한 위스키 ID

    def add_viewed_whiskey(self, whiskey_id):
        pass

    def add_to_collection(self, whiskey_id):
        pass

    def remove_from_collection(self, whiskey_id):
        pass

    def get_recently_viewed(self, count=None):
        pass

    def get_collection(self):
        pass

    def is_in_collection(self, whiskey_id):
        pass      You, 6일 전 • 뼈대 구성
```

taste_profile.py

```python
class TasteProfile:
    def __init__(self, body=3, richness=3, smoke=3, sweetness=3):
        self.body = body          # 바디감 (0-5)
        self.richness = richness  # 깊이 (0-5)
        self.smoke = smoke        # 스모키 (0-5)
        self.sweetness = sweetness # 단맛 (0-5)

    def get_vector(self):
        pass

    def __str__(self):
        pass      You, 6일 전 • 뼈대 완성
```

system.py

```python
class System:
    def __init__(self):
        self.current_user = None
        self.whiskey_catalog = Whiskys()
        self.all_reviews = {}
        self.recommendation_engine = None

        # 데이터 디렉토리 생성
        data_dir = "data"
        if not os.path.exists(data_dir):
            os.makedirs(data_dir)

    def initialize(self):
        pass

    def register_new_user(self, user_info):
        pass

    def get_current_user(self):
        pass

    def set_recommendation_engine(self, engine_type):
        pass

    def get_whiskey_details_for_display(self, whiskey_id):      You, 6일 전 • 뼈대 완성
        pass

    def create_and_add_review(self, whiskey_id, rating, text):
        pass

    def get_reviews_for_whiskey(self, whiskey_id):
        pass
```

```python
    def get_whiskey_details_for_display(self, whiskey_id):      You, 6일 전 • 뼈대 완성
        pass

    def create_and_add_review(self, whiskey_id, rating, text):
        pass

    def get_reviews_for_whiskey(self, whiskey_id):
        pass

    def get_recommendations(self, count, method_type=None, base_whiskey_id=None):
        pass

    def run(self):
        pass
```

recommendation.py

```python
class Recommendation:
    def __init__(self, user_reference, whiskeys_reference):
        self.user_reference = user_reference
        self.whiskeys_reference = whiskeys_reference

    def get_recommendations(self, count):
        pass      You, 6일 전 • 뼈대 완성
```

recommendation_similar.py

```python
import math
from recommendation import Recommendation

You, 6분 전 | 1 author (You)
class RecommendationSimilar(Recommendation):
    def __init__(self, user_reference, whiskeys_reference):
        super().__init__(user_reference, whiskeys_reference)
        self.similarity_threshold = 0.5        You, 6분 전 • Uncommitted

    def get_recommendations(self, count, base_whiskey_id=None):
        pass

    def find_similar_whiskeys(self, base_whiskey_id, count):
        pass

    def calculate_taste_similarity(self, whiskey1, whiskey2):
        pass
```

recommendation_preference.py

```python
from recommendation import Recommendation

You, 6일 전 | 1 author (You)
class RecommendationPreference(Recommendation):
    def __init__(self, user_reference, whiskeys_reference):
        super().__init__(user_reference, whiskeys_reference)

    def get_recommendations(self, count):
        pass

    def calculate_preference_match(self, user_preference, whiskey):
        pass      You, 6일 전 • 뼈대 구성
```

Ui.py

```
You, 3초 전 | 2 authors (You and one other)
class MainWindow(QMainWindow):

    def __init__(self, system_reference, parent=None):
        super().__init__(parent)
        self.system_reference = system_reference

        if hasattr(self.system_reference, 'set_ui_reference'):
            self.system_reference.set_ui_reference(self)

        self.current_selected_whiskey_id = None

        self.setWindowTitle("위스키 추천 시스템")
        self.setGeometry(100, 100, 1200, 700)

        self.init_ui()

    def init_ui(self):
        pass

    def init_details_tab(self):
        pass

    def init_preference_tab(self):
        pass

    def init_history_tab(self):
        pass

    def init_recommend_tab(self):
        pass
```

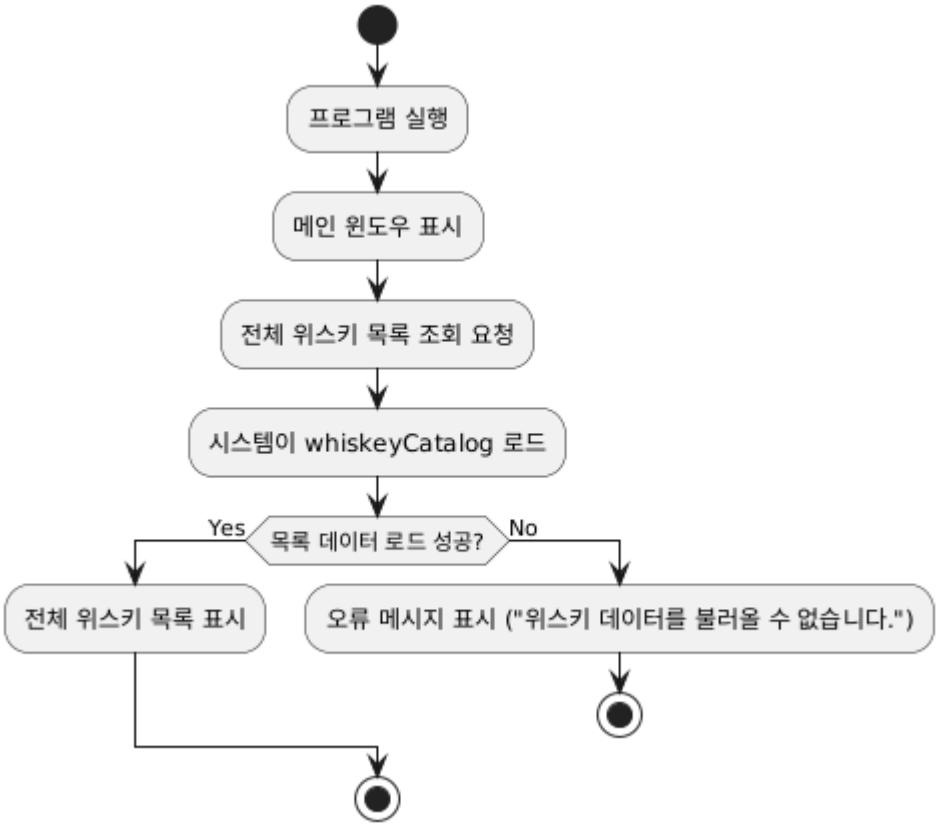
```
    def init_review_tab(self):
        pass

    def load_initial_data(self):
        pass

    def update_whiskey_list(self):
        pass

    def display_whiskey_details(self, whiskey_id):
        pass

    def display_user_preferences(self, preference):
        pass

    def update_history_display(self):
        pass

    def update_review_display(self, whiskey_id):
        pass

    def prompt_for_user_setup(self):
        pass

    @pyqtSlot()
    def on_search_changed(self):
        pass

    @pyqtSlot()
    def on_sort_changed(self):
        pass
```

```python
    @pyqtSlot()
    def on_filter_clicked(self):
        pass

    @pyqtSlot()
    def on_whiskey_selected(self):
        pass

    @pyqtSlot()
    def on_show_all_clicked(self):
        pass

    @pyqtSlot()
    def on_save_prefs_clicked(self):
        pass

    @pyqtSlot()
    def on_add_to_collection_clicked(self):
        pass

    @pyqtSlot()
    def on_get_pref_recs_clicked(self):
        pass

    @pyqtSlot()
    def on_get_similar_recs_clicked(self):
        pass

    @pyqtSlot(QListWidgetItem)
    def on_recommendation_item_clicked(self, item):
        pass
```

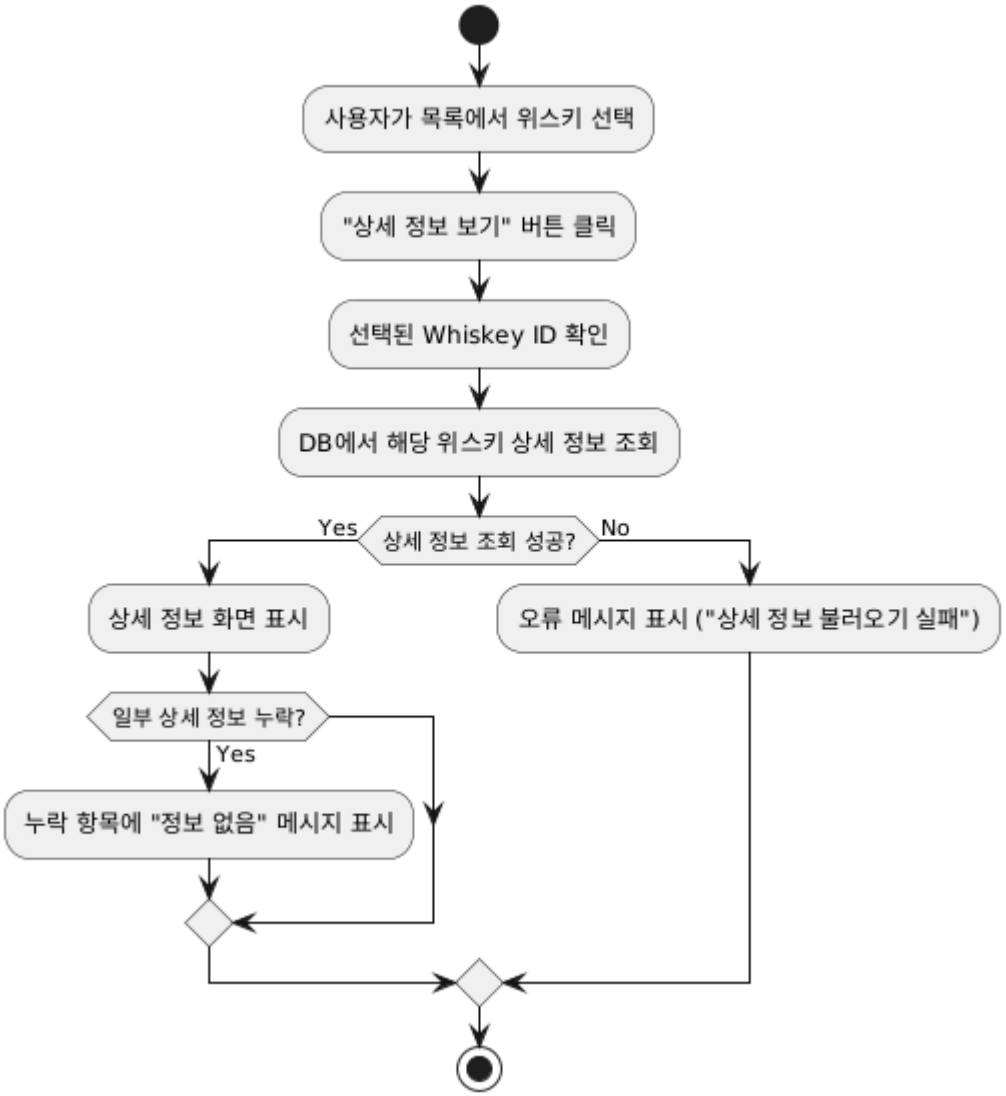
```python
    @pyqtSlot()
    def on_add_review_clicked(self):
        pass

    def closeEvent(self, event):
        pass
```

1. Activity Diagram 그리기

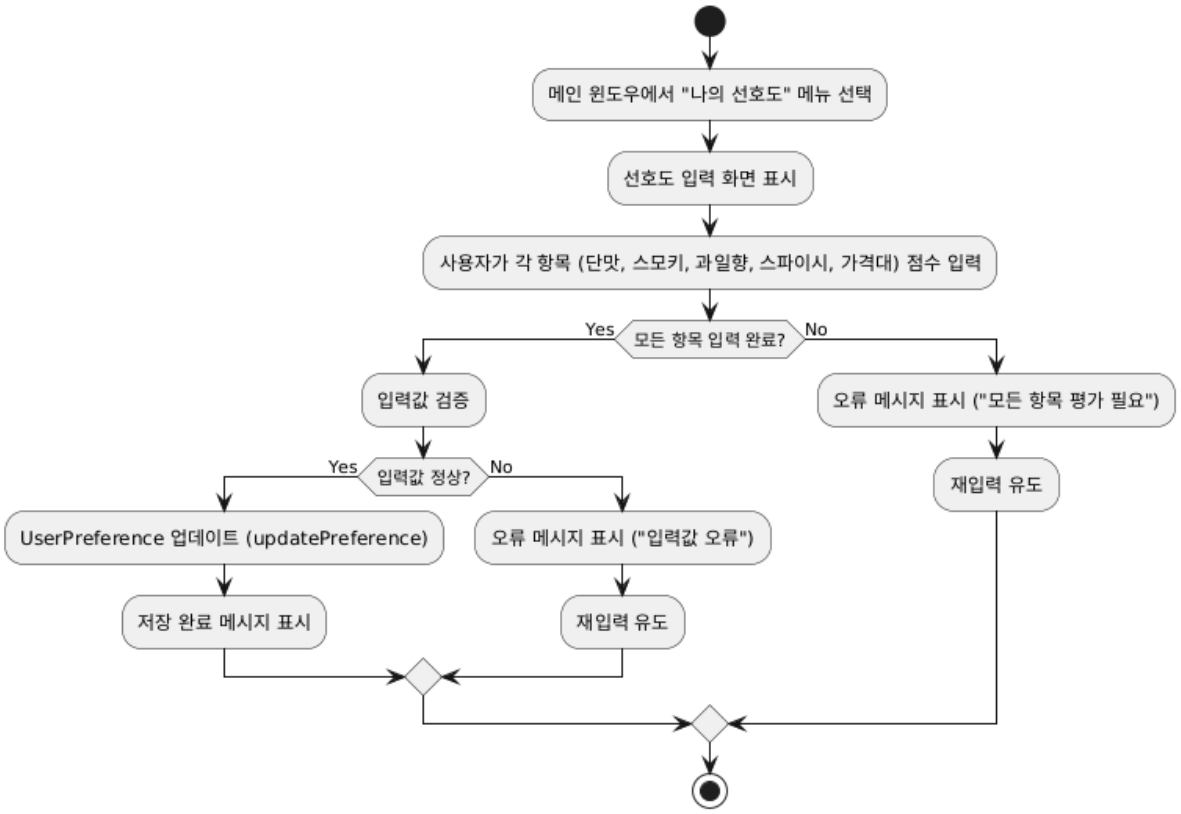
6-1. 전체 위스키 목록 불러오기


프로그램 실행
메인 윈도우 표시
전체 위스키 목록 조회 요청
시스템이 whiskeyCatalog 로드
목록 데이터 로드 성공?
Yes
No
전체 위스키 목록 표시
오류 메시지 표시 ("위스키 데이터를 불러올 수 없습니다.")

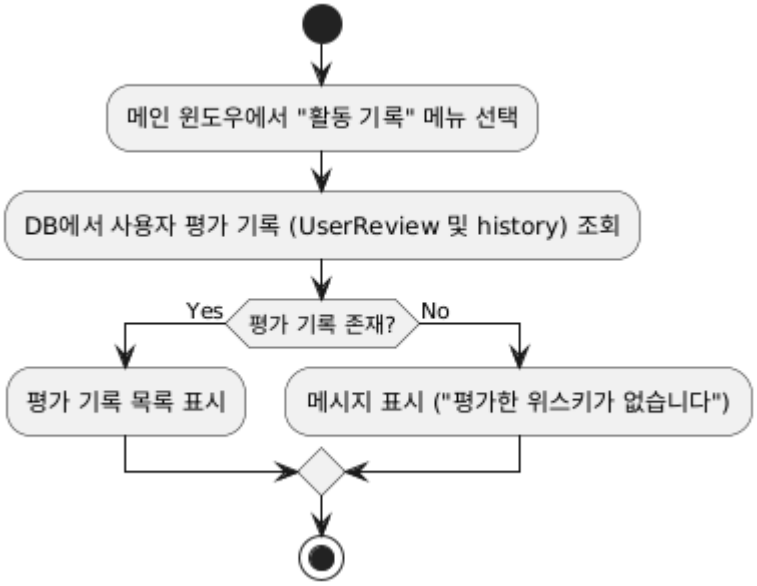
6-2. 상세정보


사용자가 목록에서 위스키 선택
"상세 정보 보기" 버튼 클릭
선택된 Whiskey ID 확인
DB에서 해당 위스키 상세 정보 조회
상세 정보 조회 성공?
Yes
No
상세 정보 화면 표시
오류 메시지 표시 ("상세 정보 불러오기 실패")
일부 상세 정보 누락?
Yes
누락 항목에 "정보 없음" 메시지 표시

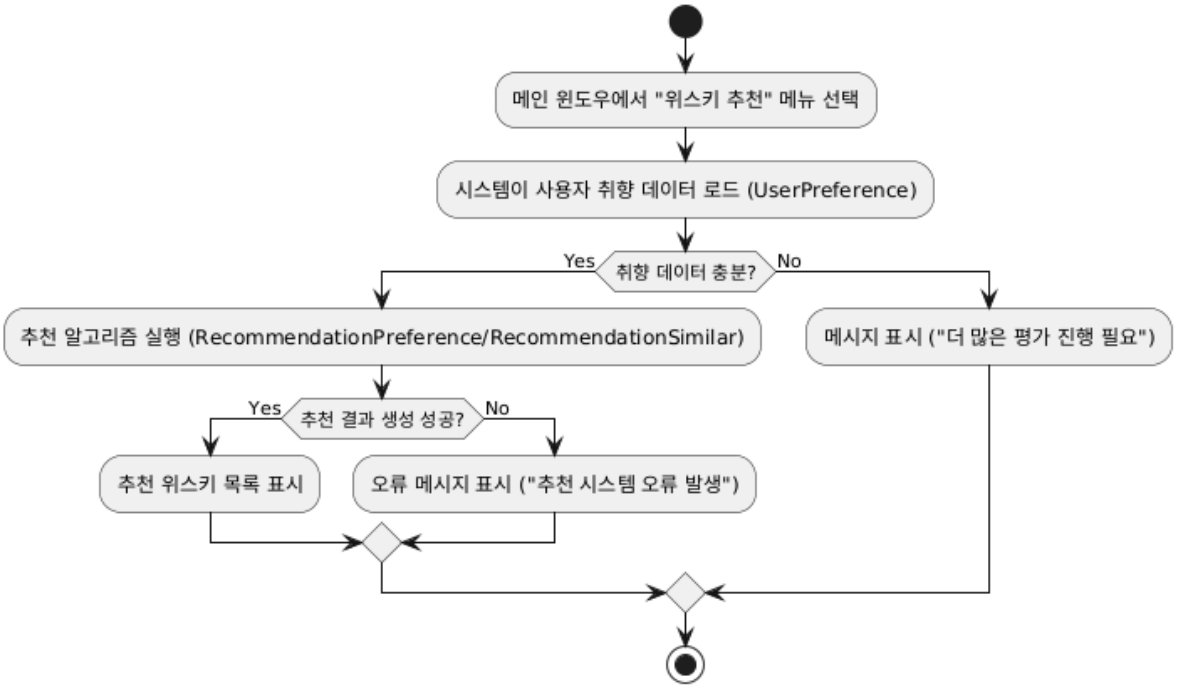
6-3. 선호도 입력



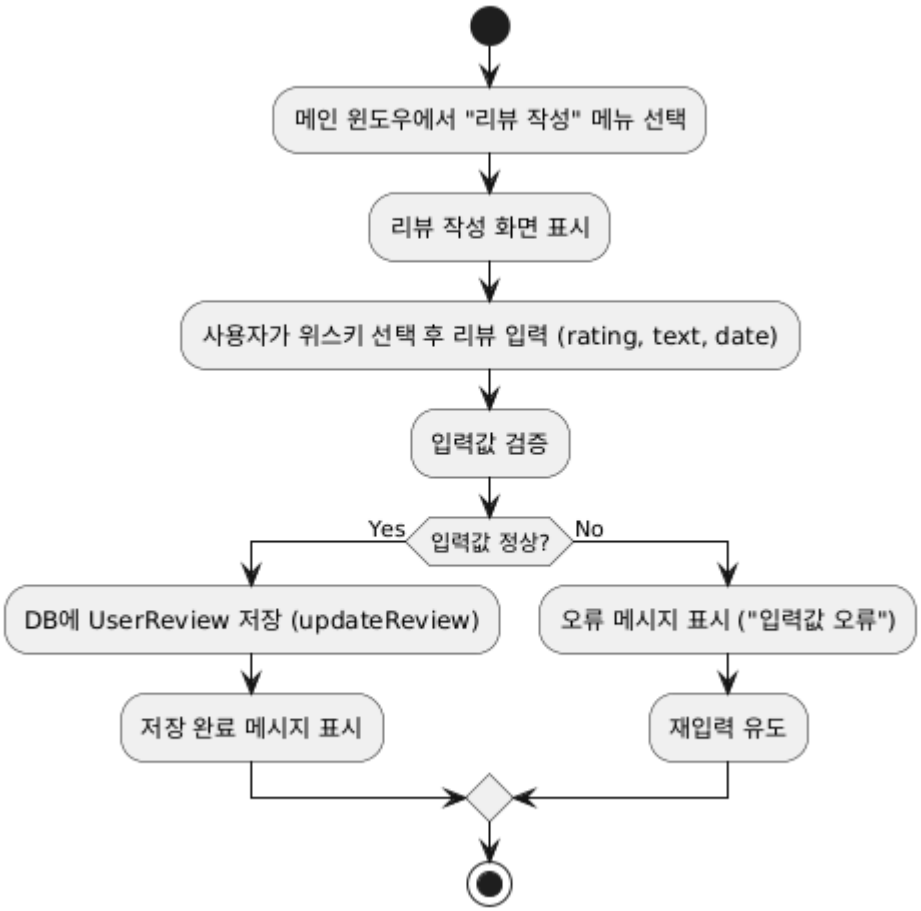
6-4. 활동 기록



6-5. 위스키 추천


메인 윈도우에서 "위스키 추천" 메뉴 선택
시스템이 사용자 취향 데이터 로드 (UserPreference)
Yes
취향 데이터 충분?
No
추천 알고리즘 실행 (RecommendationPreference/RecommendationSimilar)
메시지 표시 ("더 많은 평가 진행 필요")
Yes
추천 결과 생성 성공?
No
추천 위스키 목록 표시
오류 메시지 표시 ("추천 시스템 오류 발생")

6-6. 리뷰작성


메인 윈도우에서 "리뷰 작성" 메뉴 선택
리뷰 작성 화면 표시
사용자가 위스키 선택 후 리뷰 입력 (rating, text, date)
입력값 검증
Yes
입력값 정상?
No
DB에 UserReview 저장 (updateReview)
오류 메시지 표시 ("입력값 오류")
저장 완료 메시지 표시
재입력 유도