



목차

[1.](#_heading=h.gjdgxs) Revised Requirements 2

[1.1 Updated Requirements List 2](#_heading=h.30j0zll)

1.1.1 Natural statements on User Requirements

1.1.2 Non-functional Requirements

[1.2 Updated use case scenario 2](#_heading=h.1fob9te)

1.2.1 Specification for each requirements

[1.3 Updated Requirements Analysis 3](#_heading=h.3znysh7)

1.3.1 Requirement analysis

1.3.1.1 overall analysis domain model diagram

1.3.1.2 Briefly describe each analysis class

1.3.2 Behavioral analysis

1.3.3 Static analysis

[2. Architectural Design](#_heading=h.2et92p0)

[2.1 Identify the major components that make up the system and their interactions. 3](#_heading=h.rmayakd9b6ol)

[2.2 Add a sub-section to describe what each of the layer and major subsystems mean, their purpose etc. 3](#_heading=h.3dy6vkm)

[3.](#_heading=h.1t3h5sf) Behavioral design

3[.1 Design sequence diagram for each use case (60 ~ 100%) All DM and adequate implementation classes(including the external interface) are 3](#_heading=h.rmayakd9b6ol)

3[.2 Adequacy of method specified and correspondence to the methods in the class diagram](#_heading=h.3dy6vkm)

[4.](#_heading=h.4d34og8) Structural Design 5

4[.1 Draw a detailed design class diagram for each use case and](#_heading=h.rmayakd9b6ol) description and attributes [3](#_heading=h.rmayakd9b6ol)

4[.](#_heading=h.3dy6vkm)2 Combined class diagram for all use cases

[5.](#_heading=h.2s8eyo1) Additional contents 6

5[.1 References 3](#_heading=h.rmayakd9b6ol)

5[.2](#_heading=h.3dy6vkm) Change history(continuous from SRS)

5.3 Project status: Measurable tracking (%) against the plan Including most milestones

5.4 Team member’s contribution

5.5 Glossary

[1. Revised Requirements](#_heading=h.gjdgxs)

[1.1 Updated Requirements List](#_heading=h.30j0zll)

1.1.1 Natural statements on User Requirements

|  |  |
| --- | --- |
| No. | USER\_RS\_001 |
| Title | Create, Modify, Remove User Diary (select ver.) |
| Detail | 로그인 후 메인 화면에 나타나는 달력에서 오늘 날짜를 선택하면 다이어리를 기능이 실행된다. 일기 작성을 시작하면 사용자는 오늘의 기분이 어땠는지 묻는 질문에 ‘기쁨’, ‘우울’, ‘설렘’, ‘슬픔’ 등의 감정을 나타내는 단어를 고른다. 선택작성에서는 오늘 무슨 일이 있었는지 묻는 질문에 ‘언제, 어디서, 누가(누구와), 무엇을, 어떻게, 왜’의 질문에 맞게 사용자가 작성한다. 시스템은 사용자의 작성을 바탕으로 자동으로 산문 형식의 일기를 작성해준다. |
| No. | USER\_RS\_002 |
| Title | Create, Modify, Remove User Diary (write ver.) |
| Detail | 선택작성보다 좀 더 자세히 일기를 작성하길 원하는 사용자의 요구를 충족시키기 위해서 직접 작성 기능을 추가한다. 이때 시각화 과정을 위해서 일기에 들어간 내용을 형태소 별로 자르는 과정이 필요하다. |
| No. | USER\_RS\_003 |
| Title | Create, Modify, Remove To-Do List |
| Detail | 메인 화면에서 달력과 함께 내일 할 일을 보여준다. To-do List를 선택하면 내일 할 일을 정리하고 확인이 가능하다. 할 일을 이행하고 나서 사용자가 체크하면 해당 스케줄은 사라지고 남은 할 일들만 남게 된다. |
| No. | USER\_RS\_004 |
| Title | Exchanging Diary with friends |
| Detail | 교환일기장에 친구의 접근 권한을 부여하면 사용자와 함께 교환일기를 쓴다. 누가 먼저 쓸지, 일기 쓰는 기한 등은 두 사용자가 자유롭게 결정한다. |
| No. | USER\_RS\_005 |
| Title | Visualization and analysis User Diary |
| Detail | 일기에 들어간 내용을 형태소 별로 잘라 저장하고, 빈도수를 확인하여 일기 분석 통계에 반영한다. 이를 기반으로 일기 작성 패턴을 시각화해서 보여준다. 최근 한 달 간 많이 사용한 단어가 무엇인지 워드 클라우드 형태로 보여주고, 이때 글자 크기를 로 지정해 시각화 하여 최근 한 달 동안 어떤 감정이 주를 이루었는지 보여준다. 시스템은 일기를 언제 마지막으로 작성했는지, 얼마나 꾸준하게 작성했는지, 어떤 요일에 많이 작성했는지를 데이터 처리해준다. |

1.1.2 Non-functional Requirements

|  |  |
| --- | --- |
| No. | NF\_RS\_001 |
| Category | Product Requirement → Efficiency Requirement → Performance Requirement |
| Description | 제이콥 닐슨의 주장에 따라 사용자가 읽고 탐색하는 방식의 웹사이트는 반응시간이 1초이내를 유지해야 한다는 주장에 따라 **평균화면 반응시간을 1초이내로 설정**. 그리고 상대적으로 렌더링 시간이 긴 시각화 페이지의 경우 3초의 지연이 발생하면 사용자 25%가 웹 앱을 포기한다는 에버딘의 연구결과를 바탕으로 **일기분석 시각화 페이지 반응시간을 3초이내로 설정**.[[1]](#footnote-0) |
| No. | NF\_RS\_002 |
| Category | Product Requirement → Reliability Requirement |
| Description | 시스템은 일기 작성 및 수정, 교환 일기 등의 모든 기능에서 높은 신뢰도를 가져야 한다. **신뢰도의 기준은 시스템이 장애 없이 동작하는 시간의 비율로 98% 이상**(100번 수행했을 때 오류 동작 없이 동작하는 횟수가 98번을 넘는다.)을 목표로 한다. |
| No. | NF\_RS\_003 |
| Category | Product Requirement → Recoverability Requirement |
| Description | 시스템은 일기 작성 및 수정에서 장애가 발생하더라도 **3시간 이내에 정상 상태로 복구**되어야 한다. 시스템은 장애 복구 시간 중에 장애 상황을 공지할 수 있어야 한다. |
| No. | NF\_RS\_004 |
| Category | Product Requirement → Usability Requirement |
| Description | 회원가입을 한 사람은 자신의 일기에 대한 작성, 수정, 삭제를 자유롭게 할 수 있다. 또한 교환일기를 제외한 개인 일기는 접속한 본인만 사용 가능해야 한다. 교환일기 기능에 있어서는 교환일기 사용자 이외의 사용자는 접속할 수 없어야 한다. |
| No. | NF\_RS\_005 |
| Category | Product Requirement → Security Requirement |
| Description | **일기 전문은 데이터베이스에 인코딩을 통해 암호화 작업을 하고 저장한다**. 또한 페이지 렌더링에 사용되는 http 값 역시 **pk 값을 그대로 노출하지 않고 무작위의 코드를 사용**해 해커들이 추측성 request를 보내지 못하도록 한다. |
| No. | NF\_RS\_006 |
| Category | Organizational Requirement → Operational Requirement |
| Description | 서비스 운용에 사용자 트래픽이 증가해 성능이 감소할 때를 대비해 쉽게 확장할 수 있도록 한다. **데이터베이스와 가상서버를 따로 운용하고 컨테이너화 해서 프로젝트를 다른 서버로 이동하기 용이하게 한다.** |
| No. | NF\_RS\_007 |
| Category | External Requirement → Ethical Requirement |
| Description | 시스템 관리자 또는 개발자는 사용자의 개인 정보와 일기 데이터를 절대 동의 없이 접근 및 사용하지 않는다. |

[1.2 Updated use case scenario](#_heading=h.1fob9te)

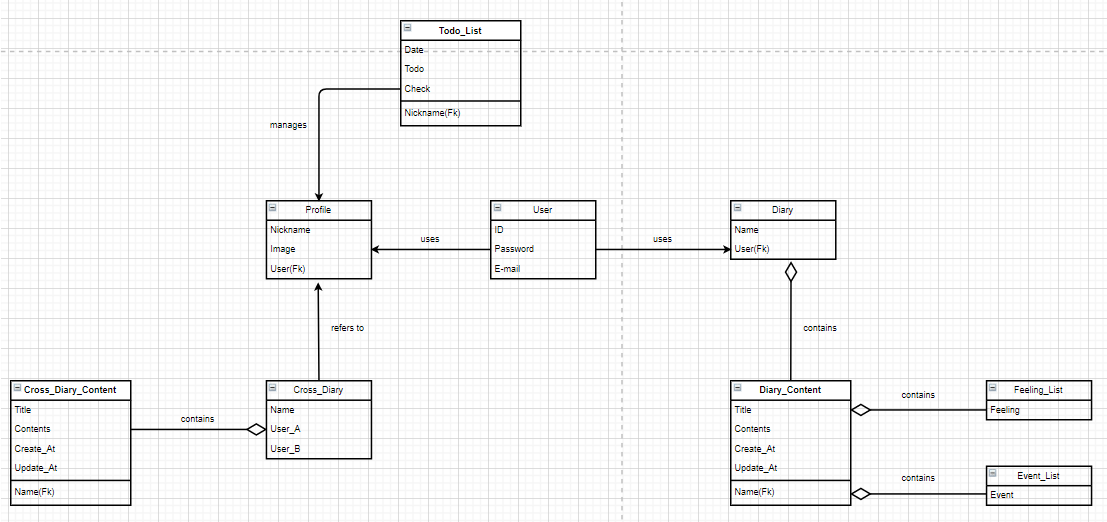
1.2.1 Specification for each requirements

|  |  |
| --- | --- |
| User case | USER\_RS\_001 |
| Success Scenario | 사용자가 원하는 일기 내용을 작성(선택)하고 내용이 데이터베이스에 성공적으로 저장되고 홈화면으로 이동 |
| Exceptions  case testing | * 사용자가 자신의 일기를 작성 완료하고 뒤로 가기 버튼(이전 페이지)를 클릭했을 때 데이터 값 저장 유무 확인 * 사용자가 선택 작성을 하는 와중에 작성을 취소했을 경우 데이터베이스에 일부 내용이 저장되는지 여부 확인 * 이미 작성되어 있는 날짜에 새로운 일기를 작성하려고 할 때 적절한 오류메시지를 출력하여 데이터베이스에 저장되는지 여부 확인 |
| User Case | USER\_RS\_002 |
| Success Scenario | 사용자가 Todo List를 작성하고 내용이 데이터베이스에 저장된다. 그 일을 마친 뒤 체크박스에 체크를 하면 데이터베이스에 달성 여부를 기록한다. |
| Exceptions  case testing | * 사용자가 Todo List를 작성하고 이미 달성된 상태로 표현이 되지는 않는지 여부 확인 |
| User Case | USER\_RS\_003 |
| Success Scenario | 교환일기를 지정한 사용자와 함께 생성하고 같은 일기에 대한 동등한 소유권을 가지며 일기 작성, 수정, 삭제를 할 수 있다. |
| Exceptions  case testing | * 두명의 사용자가 교환일기에 접근이 가능한지 또는 제3의 사용자가 접근 가능한지 여부 확인 * 두명의 사용자가 동시에 같은 날 일기를 수정했을 때 충돌이 발생하는지 여부 확인 |
| User Case | USER\_RS\_004 |
| Success Scenario | 사용자가 작성했던 일기를 바탕으로 많이 사용했던 단어나 기분, 그리고 일기 작성 횟수를 시각화 한다. |
| Exceptions  Case testing | * 사용자가 시각화 페이지를 접속할 때 결과가 3초 이내에 출력되는지 확인 * 일기 수정, 삭제 등과 같은 작업 후 방문했을 때 즉시 적용여부 확인 |

[1.3 Updated Requirements 3](#_heading=h.3znysh7)

1.3.1 Structural analysis

1.3.1.1 Domain model diagram



1.3.1.2 Briefly describe each analysis class

|  |  |
| --- | --- |
| Class Name | User |
| Description | “하루 끝”을 사용하는 사용자 |
| Relation | Association to Profile, Diary |
| Attribute | ID, Password, E-mail |

|  |  |
| --- | --- |
| Class Name | Profile |
| Description | 사용자의 개인정보 저장 |
| Relation | Association to User, Todo\_List, Cross\_Diary |
| Attribute | Nickname, Image, User(FK) |

|  |  |
| --- | --- |
| Class Name | Diary |
| Description | 사용자의 개인 다이어리를 관리 |
| Relation | Association to User, Diary\_Content |
| Attribute | Name, User(FK) |

|  |  |
| --- | --- |
| Class Name | Diary\_Content |
| Description | 사용자의 개인 다이어리의 세부 내용 |
| Relation | Association to Diary, Feeling\_List, Event\_List |
| Attribute | Title, Contents, Create\_At, Update\_At |

|  |  |
| --- | --- |
| Class Name | Feeling\_List |
| Description | 선택작성에 필요한 Feeling관련 선택목록 |
| Relation | Association to Diary\_Content |
| Attribute | Feeling |

|  |  |
| --- | --- |
| Class Name | Event\_List |
| Description | 선택작성에 필요한 Event관련 선택목록 |
| Relation | Association to Diary\_Content |
| Attribute | Event |

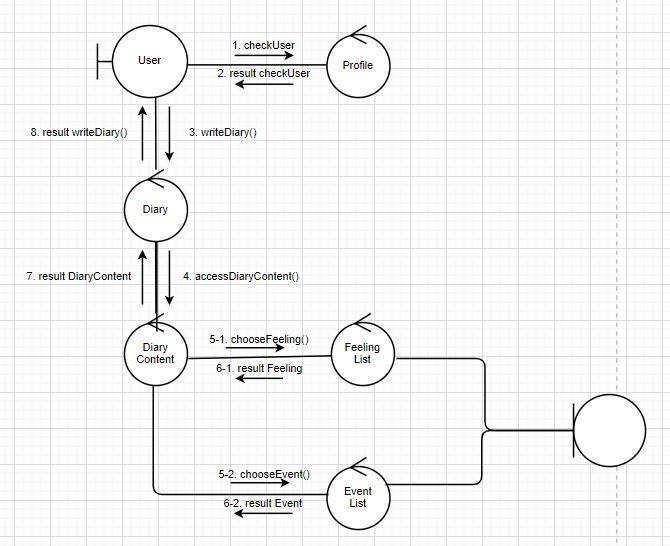
|  |  |
| --- | --- |
| Class Name | Todo\_List |
| Description | 사용자의 Todo\_List를 관리 |
| Relation | Association to Profile |
| Attribute | Date, Todo, Check, Nickname(FK) |

|  |  |
| --- | --- |
| Class Name | Cross\_Diary |
| Description | 교환일기를 공유하는 2명의 사용자 및 교환일기 관리 |
| Relation | Association to Profile, Cross\_Diary\_Content |
| Attribute | Name, User\_A, User\_B |

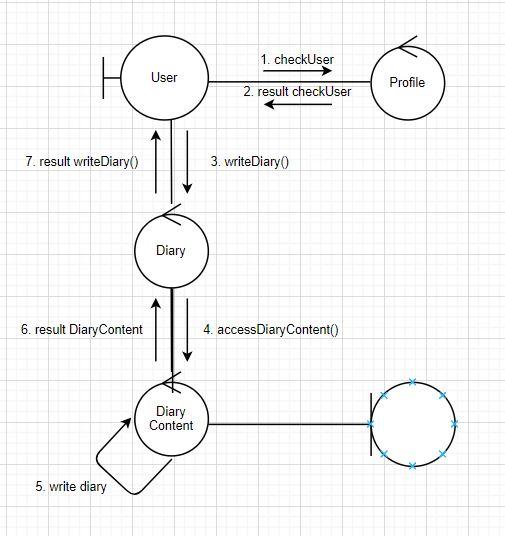
|  |  |
| --- | --- |
| Class Name | Cross\_Diary\_Content |
| Description | 교환일기의 세부내용 |
| Relation | Association to Cross\_Diary |
| Attribute | Title, Contents, Create\_At, Update\_At, Name(FK) |

1.3.2 Behavioral analysis

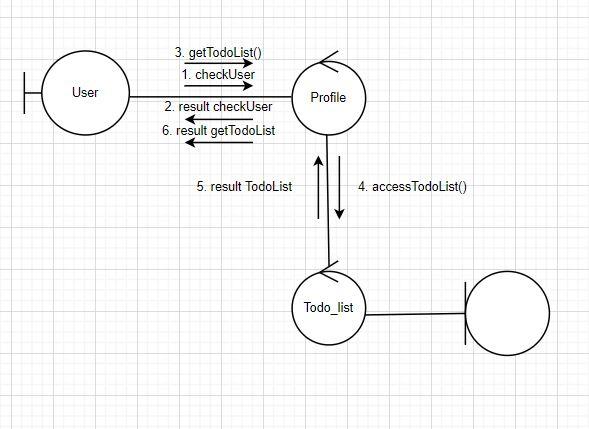
USER\_RS\_001



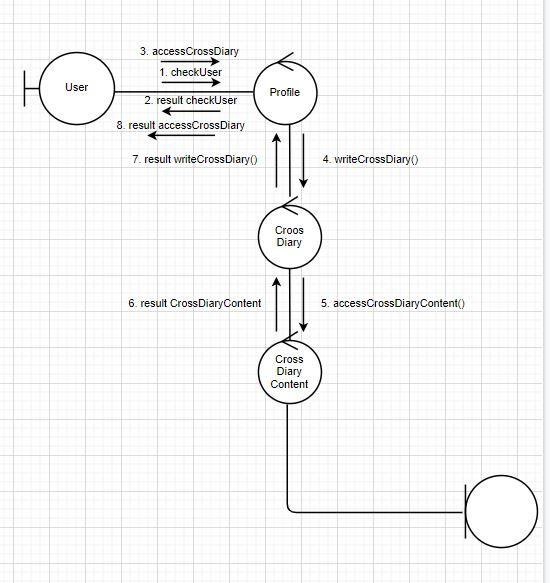
USER\_RS\_002



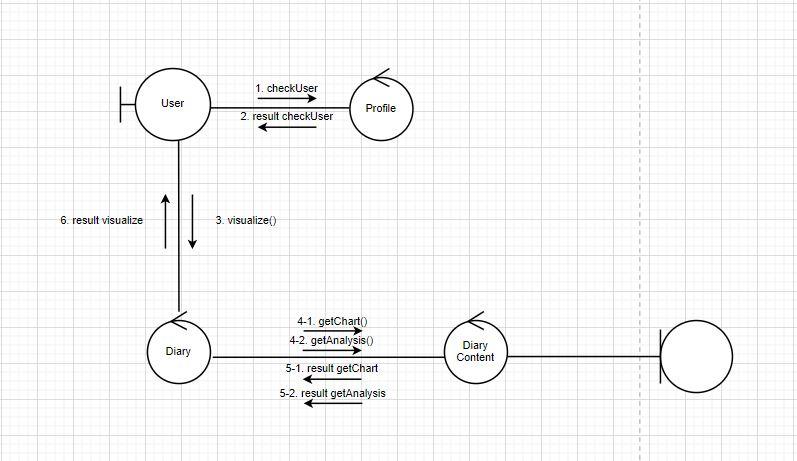
USER\_RS\_003



USER\_RS\_004



USER\_RS\_005

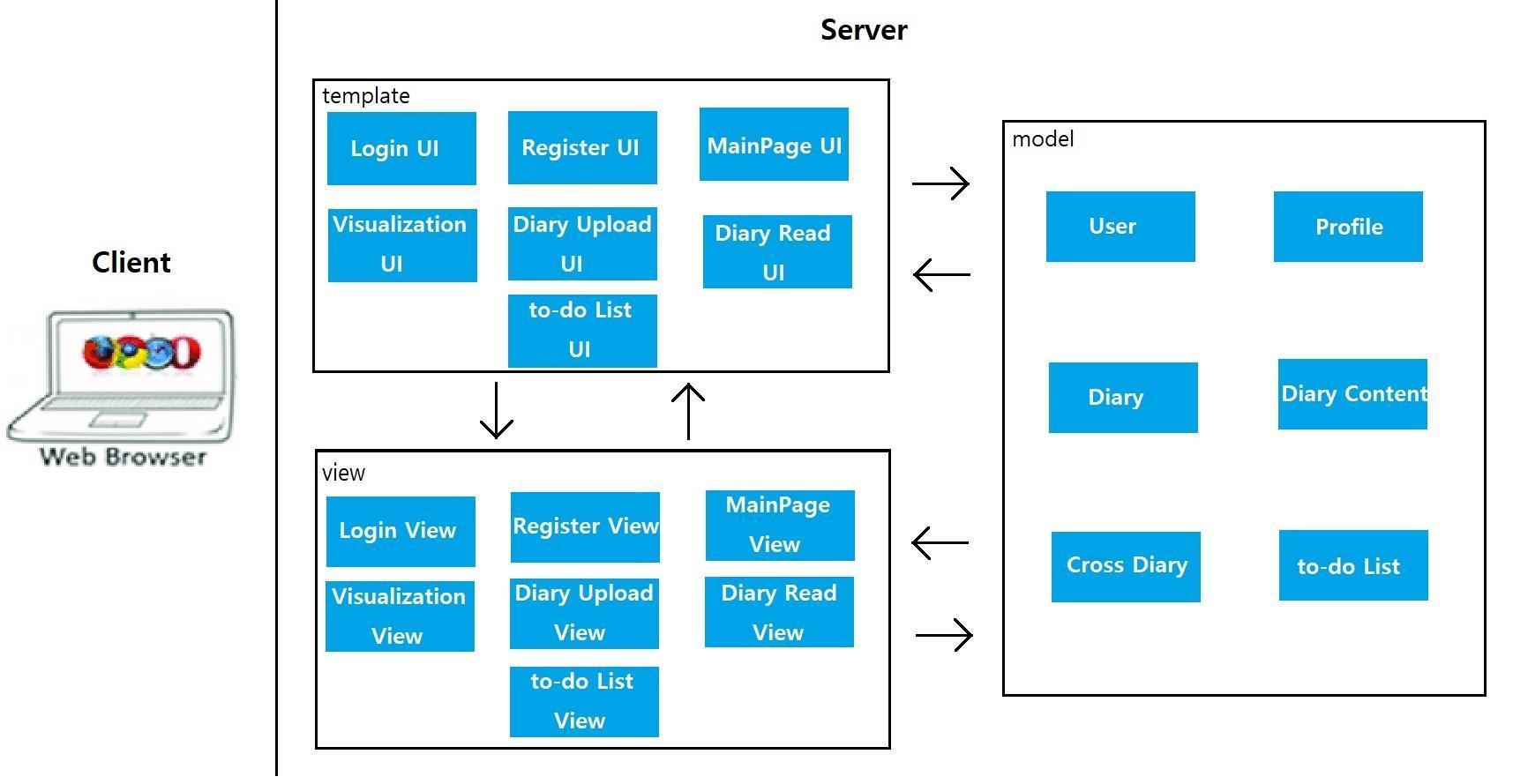


1.3.3 Static analysis

|  |
| --- |
| USER\_RS\_001 선택작성 |
|  |
|
| USER\_RS\_002 직접 작성 |
|  |
|
| USER\_RS\_003 todo list |
|  |
|
| USER\_RS\_004 교환일기 |
|  |
|
| USER\_RS\_005 작성 데이터 분석 |
|  |
|

2. Architectural Design

[2.1 Identify the major components that make up the system and their interactions.](#_heading=h.rmayakd9b6ol)



**[그림 1]**

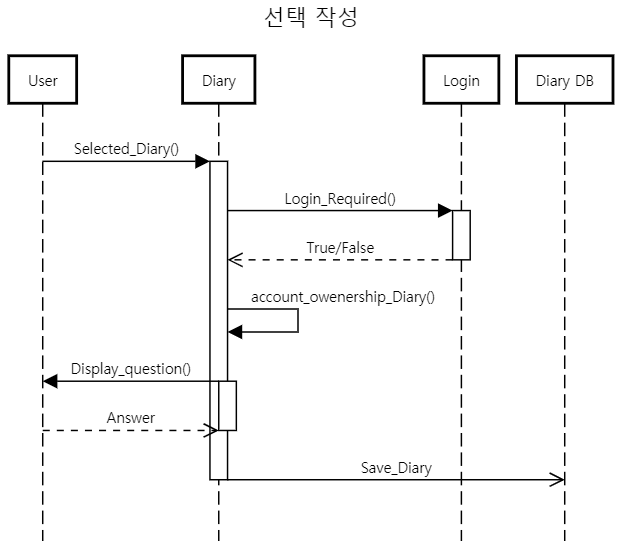
2.2

Architectural Design은 client-server model을 통해 디자인하였다. 웹 개발에 사용될 django에서는 MVT 형식을 사용해 개발을 진행하게 되는데 이는 Java spring에서 사용하는 MVC 패턴과 비슷하다. Model은 데이터베이스와 상호작용하는 인터페이스 역할이고 View는 로직을 처리하는 역할로 spring의 controller와 같은 역할이다. Template은 html과 같은 사용자에게 출력되는 화면을 나타내는 것으로 spring의 view와 같은 역할이다.

Django MVT 구조에서 View와 template은 보통 1대1로 대응되고 model은 사용자 내용을 처리하는 user, profile로 구성하고 일기 내용을 처리하는 diary와 diary\_content로 구성했다. 또한 교환일기는 cross diary로 설정했고 todo list의 기능은 todolist 테이블을 작성해 관리한다.

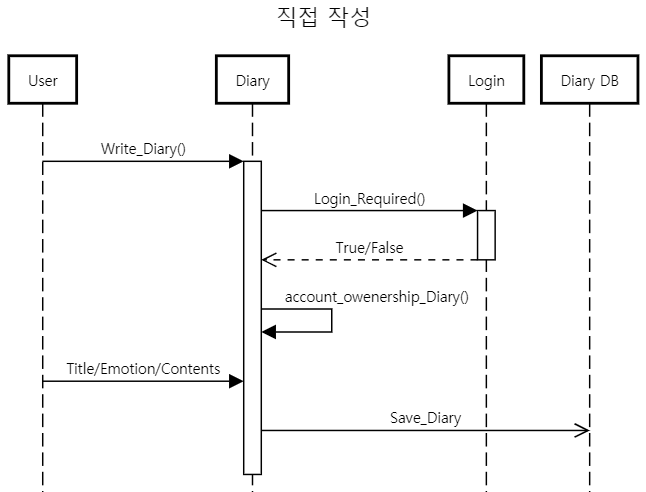
[3. Behavioral design](#_heading=h.1t3h5sf)

3.1 User\_Case\_01



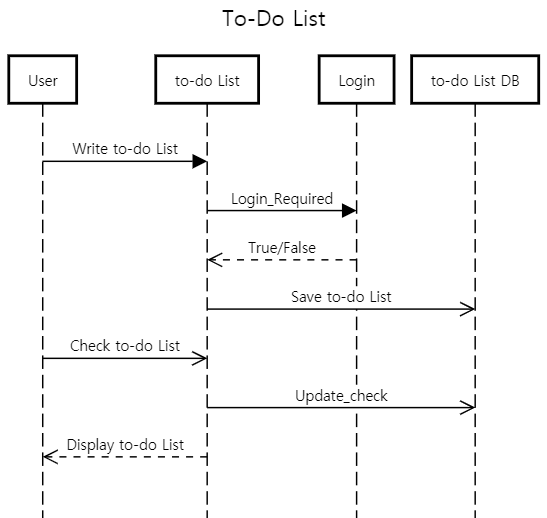
User가 Diary를 선택 작성으로 선택하면, Login 에서 로그인 여부를 확인하여 이를 Boolean 값으로 반환해준다. 로그인이 되어 있어 True를 받았다면, Diary에서 선택된 Diary에 대해 User의 접근권한을 확인한다. 접근 권한을 확인한 후에, 각 항목에 대해 Diary에서 정해진 질문을 선택 사항을 통해 제시하면 User는 이에 대해 선택하여 답하고 이를 종합하여 Diary DB에 일기 형태로 저장하게 된다.

3.2 User\_Case\_02



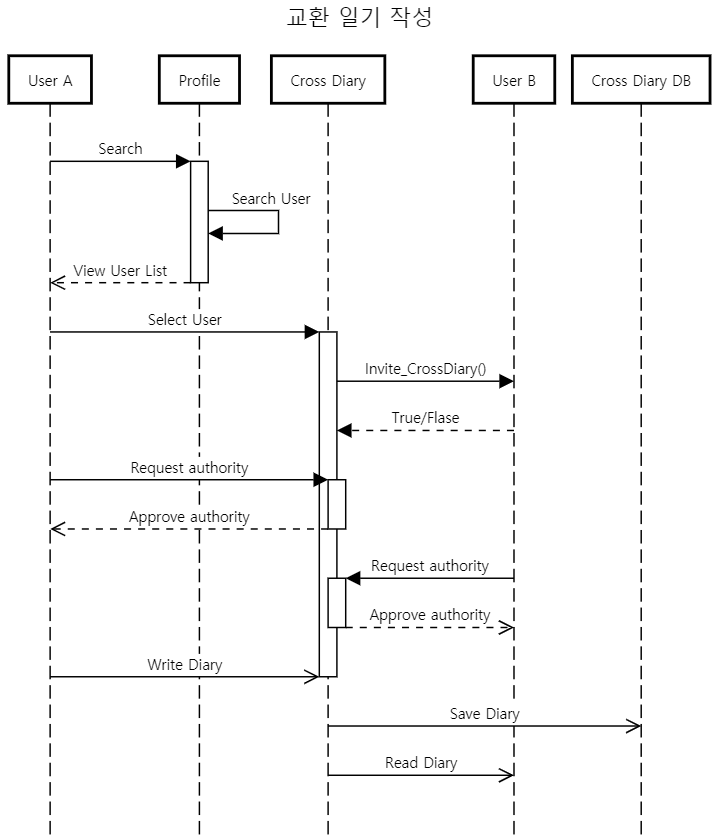
User가 Diary를 직접 작성으로 선택하면, Login 에서 로그인 여부를 확인하여 이를 Boolean 값으로 반환해준다. 로그인이 되어 있어 True를 받았다면, Diary에서 선택된 Diary에 대해 접근권한을 확인한다. 접근 권한을 확인한 후에 이 diary에 대한 대한 글을 User가 작성하게 되고, 이를 Diary DB에 일기 형태로 저장하게 된다.

3.3 User\_Case\_03



User가 to-do List를 접근하고자 하면 Login 여부를 확인한다. Login이 되어있어 True를 받았다면, 작성한 to-do List를 to-do List DB에 저장하게 된다. User는 이에 대해 수행 여부를 표시할 수 있고, 이를 to-do List DB에 반영해준다. 반영한 to-do List를 User에게 새로 보여준다.

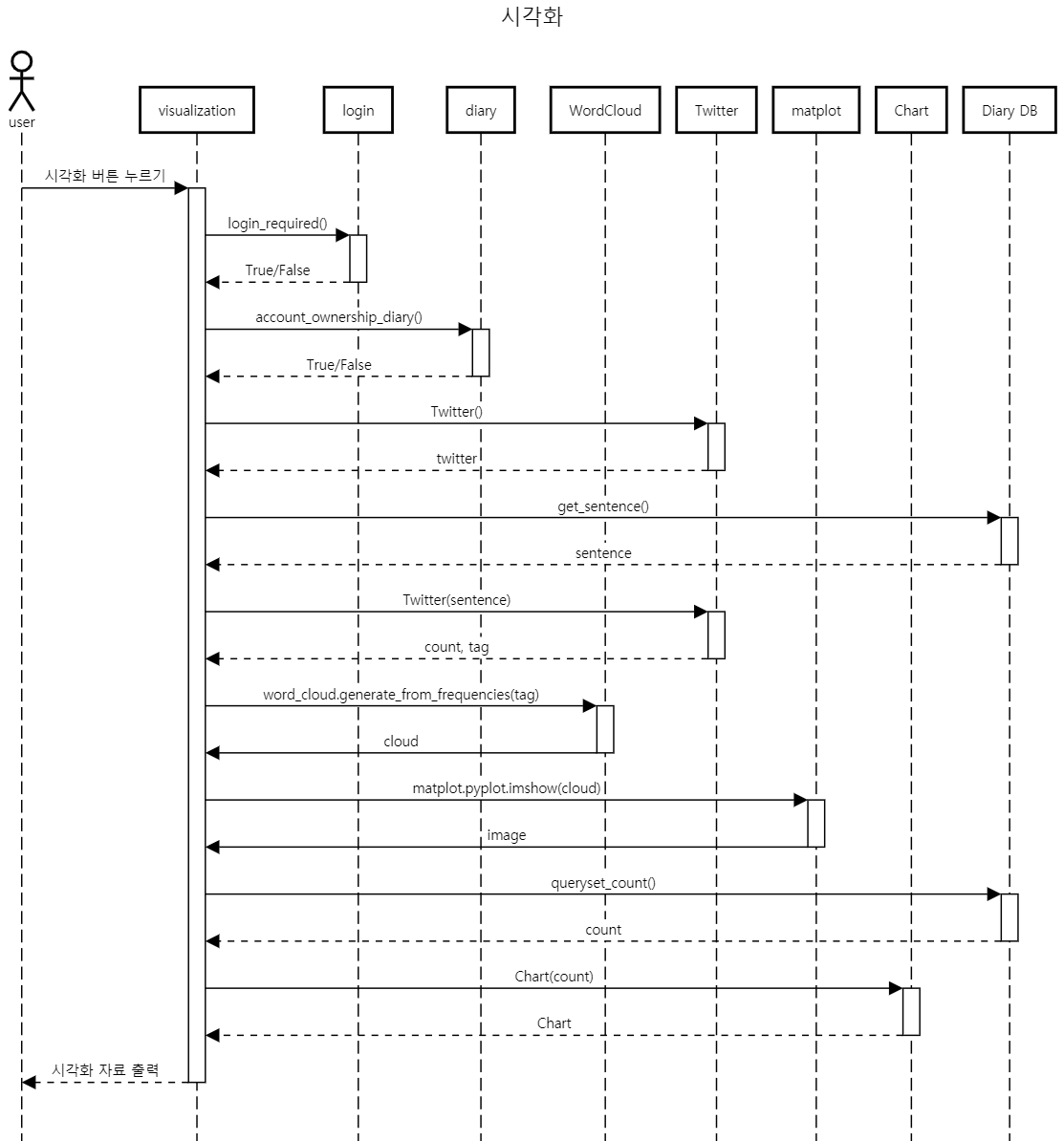
3.4 User\_Case\_04



User는 Profile을 통해 원하는 유저의 nickname 등을 검색하게 된다. User가 입력한 내용을 바탕으로 Profile에서 이를 검색해 동일한 내용을 가진 User들의 목록을 User에게 보여준다. User가 이 중 원하는 User를 선택하면 선택된 User에게 초대 메시지를 보내고, 이를 User B는 승낙/거부 여부를 대답하여 True/False값을 반환한다. True 값을 받았다면, User A와 User B는 각각 접근 권한을 자동적으로 요청하게 되고, Cross Diary는 이를 각각 승낙하게 된다.

승낙받은 User A와 User B는 각각 Cross Diary를 작성할 수 있게 되고, 만일 User A가 이를 작성하였다면 이를 Cross Diary DB에 저장하고, 이를 UserA, UserB 각각 읽을 수 있게 된다.

3.5 User\_Case\_05



User가 시각화 버튼을 클릭하면, Login 여부를 확인하여 True/False로 반환하게 된다. True를 받았다면, Diary에 대한 접근권한을 확인하여 이를 True/False로 받고 이에 대해서도 True를 받았다면 Twitter에서 Twitter 객체를 생성하여 주고, Diary DB에서 작성한 문장들을 가져온다. 이를 Twitter 객체에 넣어 형태소 별로 잘라서 tag별로 사용된 횟수를 파악하여 count에 저장하고, 각각의 tag에 대해 빈도수를 계산해 cloud객체에서 반환해준다. 이를 matplot을 통해 cloud저장된 image를 보여준다.

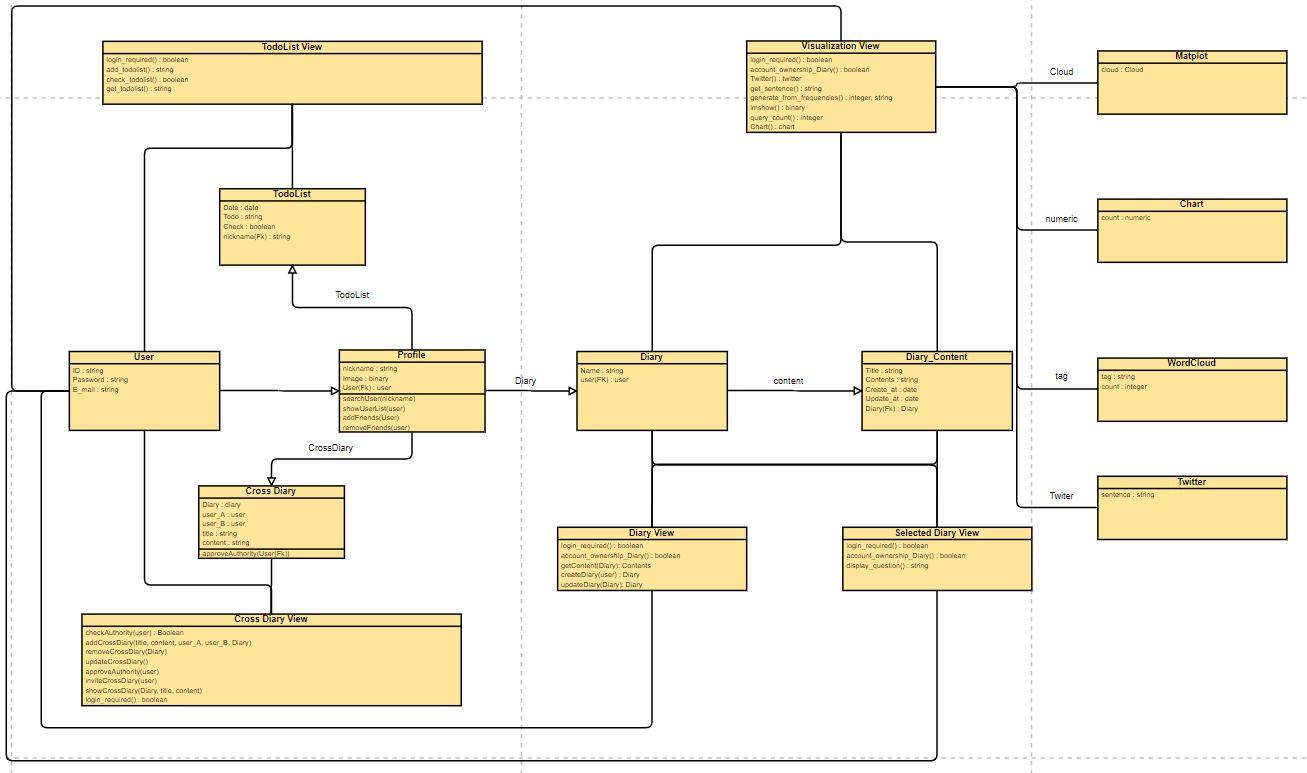
Diary DB에서 저장되었던 Feeling을 가져와서 이를 Chart형식으로 보여준다. 이를 모두 종합하여 시각화 자료로 User에게 보여주게 된다.

4. Design Structural Analysis

4[.1 Draw a detailed design class diagram for each use case and](#_heading=h.rmayakd9b6ol) description and attributes

|  |
| --- |
| 공통적으로 가지고 있는 Class   1. User : 로그인을 하기 위한 각 유저의 정보를 담고 있음. 2. Profile : 유저의 닉네임, 프로필 사진, User class를 담고 있음 . 3. Diary : 유저의 이름을 담고 있음 . 4. Diary Contents: 다이어리에 필요한 항목들을 담고 있음. |
| USER\_RS\_001 : 직접 작성 |
|  |
| Diary view  login required로 로그인 여부를 확인하고, account\_ownership\_Diary를 활용해 일기를 가져온 것에 대한 성공 여부를 판단한다.  후에 getContent로 기존 내용을 가져오고, createDiary로 새로운 일기를 추가하며, updateDiary로 기존 다이어리 내용을 수정한다. |
| USER\_RS\_002 |
|  |
| selected diary view  직접작성과 마찬가지로 login required와 account ownership으로 다이어리 사용자 여부를 판별하고, display question을 통해 diary 작성할 때에 감정 선택지를 제공한다. |
| USER\_RS\_003 |
|  |
| todolist view : login required로 로그인 판별 여부를 확인하고, get\_todolist로 기존의 todolist를 가져오고, add\_todolist()로 todo list를 추가하고, check\_todolist로 해야 할 일을 했는지 안 했는지 판별한다. |
| USER\_RS\_004 |
|  |
| cross\_diary\_view : checkAuthority를 통해 사용자 접근 여부를 판별하고, addCrossDiary, removeCrossDiary, updateCrossDiary를 통해 교환일기를 추가, 삭제, 수정한다.  cross\_diary : diary객체 하나를 생성하고, 2인이 쓰는 diary이기 때문에 userA와 userB로 사용자를 구분한뒤, 교환일기를 title, content, diary의 변수로 crossdiary를 show CrossDiary를 통해 보여준다 |
| USER\_RS\_005 |
|  |
| VisualizationView : 로그인 , 소유권 여부 판별한 뒤 twitter를 통해서 문자를 형태소 별로 잘라 문장을 분석하고, 분석할 문장들을 get\_sentences를 통해서 불러온다. 문장을 twitter로 형태소 별로 자른 다음 generate\_from\_frequencies를 통해 얼마나 많은 빈도수로 형태소가 나와있는지 빈도수를 word cloud를 통해서 데이터 화 시킨다.이것을 matplot을 통해 이미지 화 시킨 다음 imshow를 통해서 wordcloud에서 나온 이미지를 보여준다. Queryset\_count는 감정의 빈도수를 integer로 데이터 화 시켜주고, 이를 chart는 차트형식으로 그림을 보여준다. :  WordCloud : 형태소의 빈도를 통해서 도식화 하기 쉬운 데이터로 변환시킨다.  twitter : 도식화 시키기 전 글자를 형태소 별로 자른다.  chart : 감정의 빈도수를 통해서 chart로 나타낸다.  matplot : word cloud에서 가공한 데이터를 image로 변환시킨다. |

4.2



[5. Additional contents](#_heading=h.2s8eyo1)

5[.1 References](#_heading=h.rmayakd9b6ol)

1. 액티비티 다이어 그램 :

<https://online.visual-paradigm.com/app/diagrams/#diagram:proj=0&type=ActivityDiagram>

1. OTO-System Architecture : <https://www.researchgate.net/figure/OTO-system-architecture-OTO-employs-client-server-architecture-and-utilizes-the_fig6_259539890>
2. 시퀀스 다이어그램 : <https://sequencediagram.org/>
3. 클래스 다이어그램 : <https://online.visual-paradigm.com/diagrams.jsp#diagramlist:proj=0&new=ClassDiagram>
4. 참조 블로그: <https://velog.io/@namezin/%EB%B0%98%EC%9D%91%ED%98%95-%EB%94%94%EC%9E%90%EC%9D%B8-%EB%8F%84%EC%9E%85%ED%95%98%EA%B8%B0-3.-%EC%82%AC%EC%9D%B4%ED%8A%B8-%EC%86%8D%EB%8F%84-%EB%86%92%EC%9D%B4%EA%B8%B0>

5[.2](#_heading=h.3dy6vkm) Change history(continuous from SRS)

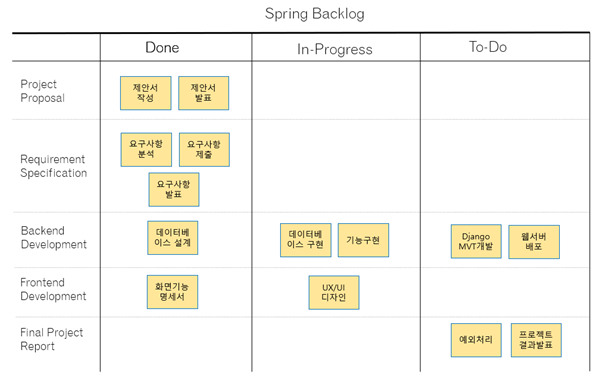
# 

|  |  |
| --- | --- |
| Previous Version | Requirement\_specification\_001 |
| New Version | Design\_Document\_001 |
| Changes | 핵심 기능 4개 → 5개  ▷ **일기 직접작성 기능**  ▷ 일기 선택작성 기능  ▷ TODO list 작성 기능  ▷ 교환일기 기능  ▷ 일기 관련 통계자료 제공 |
| Reason | 일기 작성에 자유도를 높이기 위해서, 선택작성 기능 외에 직접작성 기능을 추가하였다. |

# 

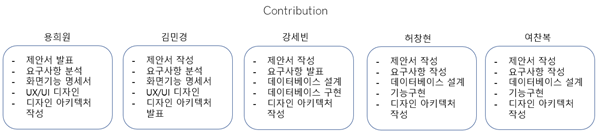
|  |  |
| --- | --- |
| Previous Version | Requirement\_specification\_001 |
| New Version | Design\_Document\_001 |
| Changes | USER\_RS별로 Activity Diagram 추가 |
| Reason | 프로세스 과정에 직관성을 더하기 위하여 USER\_RS별로 Activity Diagram을 추가하여 진행 과정을 잘 보이도록 하였다. |

5.3 Project status: Measurable tracking (%) against the plan Including most milestones



프로젝트 진행상황은 약 50% 진행되었고 본격적인 개발을 진행 중이다. 기본적인 아키텍처 및 화면 명세서 등이 완성되어 개발은 빠르고 원활하게 진행될 거라고 예상된다.

5.4 Team member’s contribution



5.5 Glossary

• 반응형 웹: 크기가 고정된 페이지가 아닌 출력되는 디바이스의 크기에 따라 인터페이스가 바뀌는 웹페이지

• SQL Query: 데이터베이스에 접근하기 위해서 사용되는 언어가 SQL을 의미하고 특정한 데이터를 조회 또는 수정을 위해 Query를 이용한다.

• HTML: 대표적인 마크업 언어로 태그로 이루어진 언어다. 웹 페이지를 작성할 때 많이 사용하는 언어

• UNIQUE KEY: 관계형 데이터베이스에서 데이터간 중복이 되지 않은 칼럼을 지칭하는 용어

• 워드 클라우드: 주로 데이터 시각화에 많이 사용되는 방법으로 단어의 빈도수에 따라 크기를 크게 한 단어 뭉치

• Entity: 개체를 의미하는 언어로 이 문서에서는 교환일기 개체 하나를 의미한다.

• 파이차트: 점유율을 보여주기 유용한 시각화 방법 중 하나이다.

• 인코딩: 사용자가 입력한 문자나 기호 등을 컴퓨터가 이용할 수 있는 신호로 만드는 것을 말한다.

• Pk: Primary key를 의미하며 관계형 데이터베이스에서 데이터에 부여되는 고유의 값이다.

• 페이지 렌더링: 서버로부터 HTML 파일을 받아 사용자에게 출력하는 과정을 의미한다.

• Http: world wide web에서 정보를 주고받을 수 있는 프로토콜이다.

• Request: 클라이언트가 웹 서버 측에 요청하는 것을 의미한다.

1. 블로그 참조 [↑](#footnote-ref-0)