1. **(a) What is SOFTWARE ENGINEERING?**

Software engineering is concerned with theories, methods and tools for professional software development. Software engineering is an engineering discipline that is concerned with all aspects of software production.

**(b) Why is it important?**

소프트웨어가 굉장히 중요한데, 소프트웨어 공학은 cost-effective하게 소프트웨어 개발을 할 수 있게 해준다. 점점 더 많은 개인과 사회가 고급 소프트웨어 시스템에 의존하고 있다. 우리는 경제적이고 신속하게 신뢰할 수 있는 시스템을 생산할 수 있어야 한다

**(c) What are differences between Software Engineering and Computer Science?**

Computer science focuses on theory and fundamentals : software engineering is concerned with practicalities of developing and delivering useful software.  
Computer Science는 이론과 근본에 중점을 둔 것이고, Software Engineering은 유용한 소프트웨어를 개발하고 제공하는 실용성에 중점을 두고 있다.

**(d) Please define “good” software. Which attributes does “bad” software have?**

Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.

* **“Good” software**: 필요한 기능(functionality)과 성능(performance)을 사용자에게 제공해야 하며 유지 보수가 가능(maintainable)하고 신뢰(dependable)할 수 있으며 사용 가능(usable)해야 한다. 즉, Maintainability (고치기 쉬워야 한다.), Dependability & security (믿을 수 있어야 한다.), Efficiency (시스템 자원을 효율적으로 사용해야 한다.), 그리고 Acceptability (사용자가 실제로 사용하기 편해야 한다.) 의 속성들을 필수적으로 가지고 있어야 한다.
* **“Bad” software**: Good software의 속성을 반대로 가지고 있는 것이다. 사용자가 원하는 기능이 없고 안 좋은 성능을 가지며 유지보수도 어렵고 신뢰할 수 없으며 사용성도 떨어지는 것이다.

**2.   (a) What is so important for software engineers to have “software engineering ethics”?**

소프트웨어 엔지니어가 전문가로서 존중받기 위해서는 정직하고 윤리적으로 책임질 수 있게 행동해야 한다.  
Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals

**(b) What are the issues of professional responsibilities?**

Confidentiality, Competence, Intellectual property rights(지적재산권), computer misuse.

**(c) How do your team apply the ACM/IEEE code of ethics into your class project?**

ACM/IEEE code of ethics :

1. Public   
2. Client and Employer  
3. Product  
4. Judgment  
5. Management  
6. Profession  
7. Colleagues  
8. Self.

우리는 우리의 프로젝트를 진행하면서 1~8을 잘 적용시키고 있다. 🡪 공공의 이익을 추구, Client와 Employer의 이익 추구, 최상의 product, 전문적 판단에서 무결성과 독립성 유지, 관리에 있어서 윤리적으로 접근, 공공 이익을 추구하며 무결성과 명성을 높임, 동료들과 공정하며 서로 지원, 끊임없는 자기발전과 윤리적 접근 추구.

**(d) How did you resolve “ethical dilemmas” in your project? Explain with an example.**

사진을 찍을 때 발생한 초상권 문제, 그냥 올리는 시스템에서 초상권 동의를 받고 올리는 시스템으로 보완.

**3.   (a) What are the software process?**

소프트웨어 시스템을 개발하는 데 필요한 일련의 활동으로, Software specification, development, validation, 그리고 evolution으로 구성된다.

* Software Specification: 고객과 엔지니어가 생산될 소프트웨어의 기능과 운영상의 제약 조건을 정의한다.
* Software Development: 소프트웨어 시스템을 설계하고 구현한다.
* Software Validation: 소프트웨어가 고객이 원하는 것을 수행하는지 확인한다.
* Software Evolution: 변화하는 고객 및 시장의 요구사항을 반영하여 소프트웨어를 변경한다.

**(b) Please compare plan-**driven vs. agile processes. Which process did your team adopt in your team project? Why?

* Plan-driven process: 모든 프로세스 활동이 사전 계획되고, 진행 상황은 이 계획과 비교하여 측정되는 프로세스이다. (waterfall, 비가역적)
* Agile process: 계획이 점진적으로 이뤄지기 때문에 변화하는 고객의 요구사항을 반영하여 프로세스를 변경하는 것이 쉽다. (Incremental, )
* 우리 조의 프로젝트는 agile process를 적용하고 있다. 초반에는 requirement 등의 개요를 정의하고 프로젝트의 중후반에는 개발을 진행하면서 점진적으로 프로젝트의 명세를 구체화할 예정이다. 현재 상황에서는 프로젝트의 가능한 모든 상황을 예측하여 계획을 완벽하게 세우기 어렵고 추후에 변경될 가능성이 있기 때문이다.

**(c) What are the problems with the waterfall model? How to overcome these problems?**

Waterfall model은 plan-driven 모델로, 명세와 개발 단계가 분명하게 분리되어 있다.

2.1.Waterfall-model.eps

Waterfall model의 가장 큰 단점은 프로세스가 진행되는 동안 변경 사항을 적용하기 어렵다는 것이다. 원칙적으로 다음 단계로 이동하기 전에 단계가 완료돼야 하기 때문이다. 따라서 이 모델은 요구사항을 잘 이해하고 설계 프로세스 진행 중에 변경 사항의 발생이 상당히 제한될 때만 적합하다. (하지만 이런 비즈니스 시스템은 거의 없다.) Waterfall model은 시스템이 여러 곳에서 개발되는 대규모 시스템 엔지니어링 프로젝트에 주로 사용된다. 이런 상황에서 waterfall model의 plan-driven 특성은 작업을 조정하는 데 도움된다.

**(d) What are the benefits of incremental development? How to apply incremental development process for your class project?**

2.2 Incremental-dev.eps

* 변화하는 고객의 요구사항을 수용하는 비용이 절감된다. 다시 분석해야 하는 요구사항과 문서화의 양은 waterfall model보다 훨씬 적다.
* 완료된 개발 작업에 대한 고객의 피드백을 얻는 것이 더 쉽다. 고객은 소프트웨어 데모에 대해 논평하고 구현된 양을 확인할 수 있다.
* 보다 신속하게 고객에게 유용한 소프트웨어를 제공하고 배포할 수 있다. 고객은 waterfall process에서보다 빨리 소프트웨어를 사용하고 이득을 취할 수 있다.

지금까지 수업을 통해 작성한 SSRD 문서와 UML diagram을 참고하여 개발을 진행하면서 프로젝트의 세세한 명세를 구체화한다. 개발은 프로토타입을 작성하여 기능과 성능 및 사용성을 검증하면서 진행한다.

**(e) Please draw the software process model (a set of activities) of your class project. Please discuss pros and cons of your process model. How to overcome the cons of your process model?**

2.2 Incremental-dev.eps

Outline description(초안/초기기획)의 내용이 완벽하지는 않아서 개발하면서 회의가 필요한 부분이 생길 수 있다는 단점이 있다. 하지만 전체적인 개요는 잘 잡혔기 때문에 세세한 부분만 조정하면 충분히 극복 가능하다. 그리고 이런 incremental development를 통해 앞으로 생길 변동에 유연하게 대처할 수 있다는 장점이 있다.

**(e) What is reuse-oriented software engineering? How to utilize them for your class project? What are their risks? How to mitigate them?**

2.3 Reuse oriented SE.eps

Reuse-oriented software engineering은 시스템의 기존 구성 요소나 응용프로그램 시스템이 통합되는 소프트웨어의 재사용을 기반으로 한다. 재사용된 요소는 사용자의 요구사항에 맞게 동작 및 기능을 조정할 수 있다. 재사용은 다양한 유형의 비즈니스 시스템을 구축하기 위한 표준 접근 방식이다.

추가) Reusable software의 유형

* 특정 환경에서 사용하도록 구성된 stand-alone application(독립 실행형 응용프로그램) 시스템.
* .Net Framework 또는 J2EE와 같은 구성요소 프레임워크와 통합될 패키지로 개발된 개체의 컬렉션.
* 서비스 표준에 따라 개발되고 원격 호출에 사용할 수 있는 웹 서비스.

우리 조 프로젝트에는 네이버 지도 API나 웹 관련 프레임워크를 재사용할 것이다. 이럴 경우 우리 프로젝트와 재사용하는 소프트웨어를 결합하면서 사용자의 요구 사항을 충족시키지 못 할 수도 있고, 재사용한 요소의 새 버전이 우리 조 프로젝트의 통제 하에 있지 않기 때문에 시스템에 대한 일부 제어가 손실될 수 있다.

소프트웨어를 재사용할 경우 발생할 수 있는 risk는 재사용할 소프트웨어를 찾고 그것을 이해해서 프로젝트에 적용해야 하는 부담이 있고, 재사용된 소프트웨어의 소스코드나, 툴이 기존의 프로젝트와 호환될 수 없다면 보수 비용이 더 높아질 것이다. 이를 완화시키기 위해서는 재사용할 소프트웨어와 관련된 requirement의 정제 단계를 소프트웨어 엔지니어링 프로세스에 추가하고, 재사용할 소프트웨어 후보를 찾고 평가하는 단계도 명시적으로 추가해야 한다.

**(f) What is V-model? Please suggest the V-model for your class project with detailed milestones (e.g., deadline of each activities; deliverables).**

V-model은 plan-driven software process의 testing 단계에 사용되는 모델로, waterfall model의 각 단계에 해당하는 소프트웨어 유효성 검사 작업을 보여준다. 모델을 옆으로 돌려서 보면 V모양이어서 V-model이라고 한다.

우리 프로젝트의 V-model

2.7 Testing-phases.eps

**(g) How to make the evolution plan of your class project? What are the top 3 risks of your class project due to evolution? How to mitigate them?**

우리 조 프로젝트의 주요 기능은 아니지만 있으면 편리한 기능들이나 프로젝트 규모가 커져야 가능한 기능을 evolution plan으로 만들었다. 사진에 해쉬태그 기능 추가나 관리자 인터페이스 추가 및 광고 창 제작 등이다.

[재작업 비용]

* 변경되거나 새로운 시스템 요구사항이 생기기 때문에 요구사항을 재분석해야한다.
* 새로운 기능을 구현해야 한다.
* 플랫폼을 변경하려면 애플리케이션도 변경해야 한다.

[Risk를 완화시키는 방법]

* 상당한 재작업이 요구되기 전에 가능한 변경을 예상할 수 있는 활동을 소프트웨어 엔지니어링 프로세스에 포함해야 한다. 예를 들어, 프로토타입 시스템은 고객에게 몇 가지 핵심 기능을 보여주기 위해 개발될 수 있다.
* 비교적 낮은 비용으로 변경 사항을 수용할 수 있도록 프로세스를 설계한다. 이것은 incremental development의 형태를 포함한다. 제안된 변경 사항은 아직 개발되지 않은 increments로 구현될 수 있다. 이것이 불가능할 경우 변경 사항을 통합하기 위해 시스템의 작은 부분(single increment)만 변경할 수 있다.

**4.   (a) Is ‘extreme programming’ good or bad for your project?  Justify your answer.**

GOOD.  
1. 작은 규모의 개발팀  
2. 고객과 팀원 간의 대화 가능  
3. 모두 컴퓨터학과로 구성 -> pair programming 용이.

**(b) What are output of ‘extreme programming (XP)’ (i.e., results produced by XP)?**

자체적인 테스트를 통과한 새로운 릴리즈 버전의 프로그램

**(c) Please explain “Scrum Sprint Cycle”? How can you apply it for your class project?**

2~4주 정도의 짧은 주기를 두고 잦은 미팅을 가짐으로써 매 회의마다 유의미한 개발 결과물을 도출해내는 과정.

우리도 자주 만나서 맨날 뭔가 만든다. 정기적인 회의를 통해 유의미한 결과물 도출.

**(d) What is ‘pair programming’? What are its pros and cons?**

개발자들이 짝을 이루어 서로의 작업을 체크하고 지원을 주고받으며 개발을 수행해 좋은 성과를 내는 것. 짝은 동적으로 바뀌어 개발기간 동안 모든 팀 멤버와 한 번씩 같이 작업한다.

장점 : 공동의 책임감과 주인의식을 부여한다. / 비공식적인 피드백 과정으로 작용한다. 그에 따라 소프트웨어의 에러를 발견하는데 효율적이다. / 소프트웨어의 구조를 발전시키기 위한 재구성에 도움이 된다.

단점 : 정식으로 코드 검사를 진행하는 것보다 에러 비율이 높다. BUT, 비용적 측면에서 이득.

생산성적인 측면에서 전문가끼리 짝을 맺는 것 보다 학생 지원자와 전문가가 짝이 되는 경우가 더 높은 효율을 보였다.

**5.   (a) What is a requirement? Why is it so important to have right requirements in early stage of software development?**

어떻게 운영되고 어떻게 개발될 지에 대한 고객들의 요구사항 또는 제약사항을 구체적으로 정의한 것.

Requirement: the service that a customer requires from a system and the constraints under which it operates and is developed.

처음에 고치는 것이 나중에 고치는 것보다 비용이 훨씬 적다.

It is important to have right requirements in early stage of software development because fixing a requirements error after delivery costs much higher than the cost of fixing an implementation error

**(b) What are Key challenges of RE? How to overcome them?**

RE = Requirement Engineering

Key Challenges : 완전하고 일관된 요구사항을 정의하는 것.

Requirement should be both complete and consistent, but because of system and environmental complexity, it is impossible to produce a complete and consistent requirements document.

Validity, consistency, completeness, realism and verifiability 를 수시로 체크해야한다. 이는 requirements reviews, prototyping and test-case generation를 수행함으로써 만족시킬 수 있다.

To overcome this problem, requirements validation should be done strictly so that it maximizes the completeness. Validity, consistency, completeness, realism and verifiability are the conditions to be checked. It can be validated by requirements reviews, prototyping and test-case generation.

**(c) What are five key failure factors related to RE among CHAOS projects (See the WinWin slides)**

**(d) What are functional and non-functional requirements? Why are non-functional requirements hard to elicit? How did you find them in your class project?**

Functional requirements: how the system should react to particular inputs and how the system should behave in particular situations. / 우리 시스템의 기능에 대한 요구사항.

Nonfunctional requirements: constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc. / 기능 이외에 성능(performance), 수행시간(runtime) 등에 대한 요구사항.

Nonfunctional requirements are hard to elicit, because NFR often apply to the system as a whole rather than individual features or services i.e a single non-functional requirement may generate a number of related functional requirements that define system services that are required.

이유 : 개념 자체가 원래 모호하다. Functional은 구체적이지만, non-functional은 추상적인 것이 많다. 특정 기능이 아니라 전체 시스템에 관련되어 있기 때문에 정의하기 어렵다.

예를 들면, 사진을 주고받는 서비스 특성상, 초상권 보호가 중요한 관점이기 때문에 non-functional requirement중 하나로 보안을 설정하였다

**(e) What to describe in requirements document? (See SSRD in the MBASE slides)**

A requirements document explains why a product is needed, puts the product in context, and describes what the finished product will be like. A large part of the requirements document is the formal list of requirements.

요구사항, 왜 해당 제품/서비스가 필요한 지, 구체적인 결과물에 대한 예측이 들어가야 한다.

The document should include an **overview of the finished product** in narrative and/or **graphical mockup**(model) form.

이 문서는 완제품에 대한 간략한 설명과 그에 대한 시각적 이미지를 포함해야 한다.

**(f) How to validate requirements? How do you know whether your requirements are correct** **and whether your requirements are correctly described?**

- requirements reviews; systematic manual analysis of the requirement

- prototyping; using an executable model of the system to check requirements

- test-case generation; developing tests for requirements to check testability

**(g) What are types of requirements specification? What are advantages and disadvantages of them?**

(1) natural language – 줄글?  
(2) structured natural language - SSRD에서 쓴거. 표?   
(3) design description languages – Programming Language?? Interface Specification에 쓰인다.  
(4) graphical notations – 그림?  
(5) mathematical specification – 수치 도표?

<Natural language>

Pros. Natural language is very expressive, intuitive, and universal; which means requirements can be understood by users and customers.

Cons. Lack of clarity, requirements confusion, requirements amalgamation are disadvantages of natural language.

<Structured natural language>

Pros. Structured specification works well for some requirements(like requirements for embedded control system).

Cons. However, it is too rigid for writing business system requirements.

**(f) How to represent requirements consistently, completely, and non-ambiguously? How does the software industry do for this goal? – 위에 f와 동일한 답.**

You need to ask these questions to represent requirements consistently, completely, and non-ambiguously. For consistency, ‘are there any requirements conflicts?’. For completeness, ‘are all functions required by the customer included?’. For non-ambiguity, ‘do you describe your terms in a clarified way?’.

**(e) What are ‘good’ requirements? What are ‘bad’ requirements? Please provide three examples of ‘bad’ requirements specification in the SSRD (requirement document) of your class project. Please correct them into ‘good’ requirements.**

To make good requirements, it needs to be as specific as possible. **Consistency, completeness, non-ambiguity are also necessary elements**. If these elements not included, it will be bad requirements.

Consistency: model, user, a person taken by, photographee등 용어가 산발. 용어를 하나로 일치시켜서 ssrd에 통일된 용어를 사용할 것.

Completeness: 앱의 특성 상 작가들의 사진을 보는 기능이 필요한데 그런 기능이 requirement에 없었다. 그래서 ‘view photo’의 기능을 추가하였당.

Ambiguity: rating이 어떻게 산출되는지 모호했다. 그래서 구체적으로 rating이 이루어지는 방법 à like + grading = rating

**6.   (a) What is MBASE? How can it help your class project?**

Process, product, property를 종합하는 소프트웨어 공학 접근법. Winwin spiral 접근법을 통해서 OCD,SSRD,SSAD,LCP,FRD 를 계속적으로 개선해나가면서 시스템을 개발하는 것이 본질.

SSRD를 작성함으로써 우리의 아이템에 대한 구체적인 요소들에 대해 정의할 수 있었다.

**(b)   Describe the WinWin negotiation results of your class project. What are pros and cons of the WinWin approach? How can you get benefits for your project practically from WinWin?**

Not position!! Interest를 봐야한다!

장점 : stakeholder들의 요구사항을 최대한으로 만족시켜서 당사자들이 모두 결과에 만족할 수 있다.

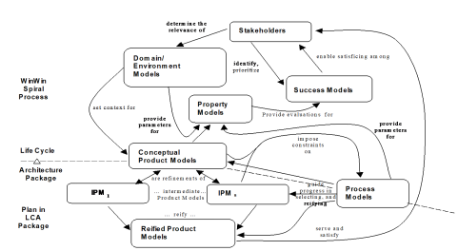
단점 : 상대방의 interest에 대한 이해가 부족하면 효과적이고 만족스러운 결과를 기대할 수 없다.

* Win -1 : 다른 사람들의 기호를 최대한 쉽게 파악하고 사진을 보고 싶다.
* Win-2 : 사진작가들의 실력을 증명하고 싶다
* Win-3 : 다양한 사람들이 다양한 장소에서 찍은 사진을 검색해서 보고싶다.
* Win-4 : retouching 을 통해서 사진을 완벽하게 만들고 싶다
* Win-5 : 자신이 원하는 특정 장소에서 사진을 찍고 싶다
* Win-6 : 모든 과정을 끝내고 photographee에게 사진을 보냈을 때 만족도를 grading하고 싶다
* Win-7 : 좋은 장비로 좋은 장소를 사진찍고 싶다
* Win-8 : photographer의 정보를 확인하고 싶다(믿을만한지…)
* Win-9 : photographee가 photographer의 profile을 확인하고 싶다.
* Win-10: photographer와 phorographee가 소통을 할 수 있었으면 좋겠다.

**(c)   Apply MBASE (especially SSRD) for your class project. Find out mismatch of models (or their examples of the mismatches) in MBASE.**

Photographee는 비용을 지불하는 것을 원하지 않고 photographer는 수익을 얻고 싶어함

**(d)   Explain how the MBASE conceptual framework (shown in the following picture) work using your class project**



1. Success Model을 먼저 결정할 수 있다.
2. 이를 성공하면 Product가 알 수 있다..
3. 어떤 기능이 있어야 하는 지가 보이고, = 그 Concept을 알 수 있음.
4. How fast(개발 예상 소요기간) 알 수 있다..
5. 코드를 어떻게 짜야 하는 지도 알 수 있다..

**(e)   What are elements of critical front end milestones? What are different between LCO and LCA? Why is Life-cycle plan necessary?**

1) Definition of operational concept, Definition of system requirements, definition of system and software architecture, definition of life-cycle plan, Feasibility rationale

2) LCO LCA 비교.

\* Life Cycle Objectives (LCO):

1. **less structured**, with information moving around
2. focus on the strategy or "vision" (e.g., for the Operational Concept Description and Life Cycle Plan), as opposed to the details
3. could have some mismatches (indicating unresolved issues or items)
4. no need for complete forward and backward traceability
5. may still include "possible" or "potential" elements (e.g., Entities, Components, …)
6. some sections could be left as TBD, particularly Construction, Transition, and Support plans

\* Life Cycle Architecture (LCA):

1. more **formal**, with solid tracing upward and downward
2. no major unresolved issues or items, and closure mechanisms identified for any unresolved issues or items (e.g., “detailed data entry capabilities will be specified once the Library chooses a Forms Management package on February 15”)
3. no more TBDs expect possibly within Construction, Transition, and Support plans
4. basic elements from the Life Cycle Plan are indicated within the Construction, Transition, and Support plans
5. there should no longer be any "possible" or "potential" elements (e.g., Entities, Components, …)
6. no more superfluous, unreferenced items: each element (e.g., Entities, Components, …) either should reference, or be referenced by another element. Items that are not referenced should be eliminated, or documented as irrelevant

3) Identification of the major stakeholders now, future

현재와 미래의 주요한 이해관계자들을 식별하는 데 필요하다.

**(f)   Please apply “feasibility rationale” for your class project. What can the reader get out of this ‘feasibility rational’ document?**

Feasibility rationale = 만들고자 하는 기능에 대해 예측할 수 있는 Risk를 정의하는 것.

가정: 유저들은 서로 채팅을 해야한다. 사진을 업로드할 수 있어야하고, 사진의 정보를 입력할 수 있어야한다. 사진, 유저와 메시지DB를 servlet(통신 프로토콜..?)을 통해 전송할 수 있어야한다..

Risk : 시간,프로그램이 제 시간에 구현하기에 너무 많은 기능들이 있다. / Lack of knowledge

**7.   (a) What is UML? How can it help your class project?**

UML=Unified Modeling Language로 통합모델링 언어이다. 시스템의 전체모습, 큰그림(청사진) 등, 즉 아키텍처를 보여주는(나타내기 위한) 표준적 표기 언어이다. 구조(class), 상호작용(Sequence)등을 구조적으로 그림을 통해 표현한다. 🡪 말 그대로 우리 아이템의 시스템의 전체적인 모습을 효율적으로 쉽게 볼 수 있다.

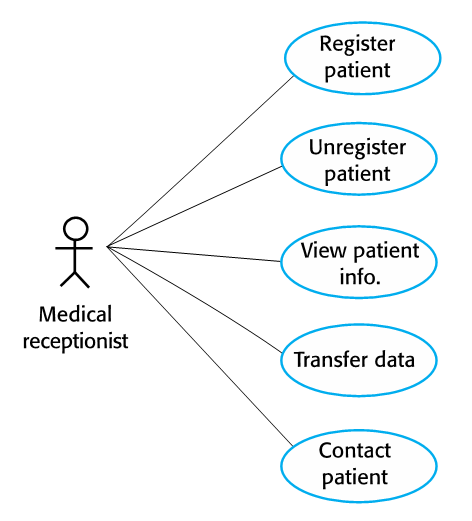
**(b)   Explain the following diagram (what it is, why important, how to draw)**

A.   Use case diagram

WHAT : 시스템과 시스템의 환경에서의 상호작용을 보여주기 위한 다이어그램(Interaction Modeling로 쓰일 수 있다)

WHY : USE Case 다이어그램은 본디 요구사항의 온전한 도출을 하기 위해 쓰이며 다이어그램은 사용 사례에 대한 개요와 **텍스트 폼에서보다 더 구체적인 설명을 제공**하기에 중요하다.

HOW to Draw



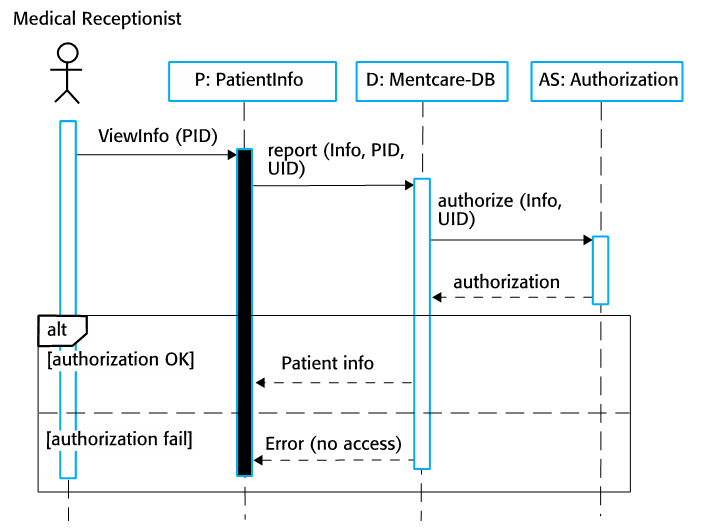
B.   Sequence diagram

WHAT : 시스템과 사용자, 시스템의 구성요소(시스템 기능)간의 상호작용을 보여주기 위한 다이어그램(Interaction Modeling로 쓰일 수 있다)

WHY : Actor(시스템이나 사람)과 시스템 내의 개체와의 상호작용을 보여주고 Use Case에서 명시된 시스템과의 상호작용 또한 보여주기 때문에 중요하다

How Draw :

1. 관련 객체와 시스템(Actor)는 다이어그램의 맨 위를 따라 나열되고, 그에 수직으로 점선을 그린다  
2. 개체간의 상호작용은 주석을 단 화살표로 나타낸다  
3. 쓰는 순서는 (1)위에서 아래로 (2)왼쪽에서 오른쪽으로 쓴다

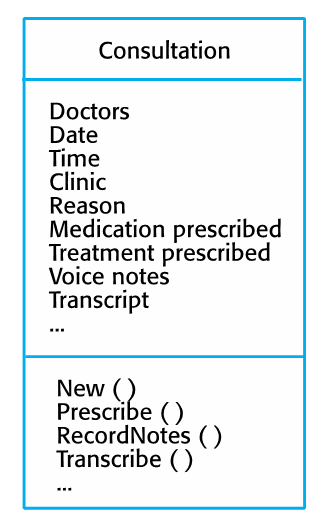


C.   Class diagram

WHAT : 시스템 대상 구조와 이들 구조 간의 관계를 보여주는 다이어그램

WHY : 시스템 구조는 시스템 객체의 일반적 정의로 생각할 수 있다. 때문에 이 다이어그램은 구조들간의 관련성과 시스템내의 구조를 보여주는 객제 지향 시스템 모델을 발전시킬 때 사용할 수 있다.

How Draw : 큰 상자를 그리고 삼단으로 나누어 맨 위에 consultation, 중간 칸에 object, 마지막 칸에 behavior를 넣는다.

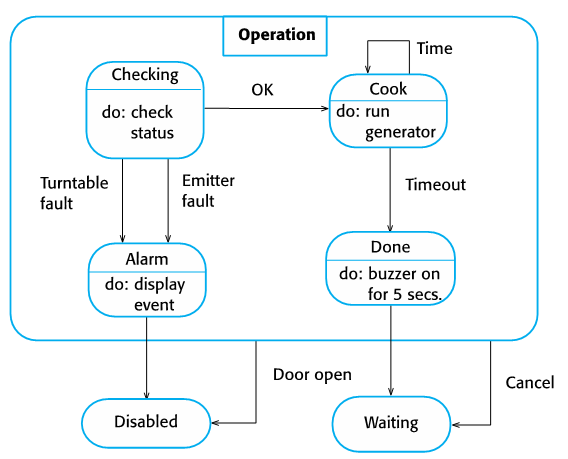


D.   State diagram

WHAT : 시스템이 어떻게 내부적, 외부적 활동에 반응을 하는지를 보여주는 다이어그램

WHY : 시스템의 자극에 대한 반응성을 보여주므로 모델링에 있어서 실시간 시스템을 모델링할 때 쓰인다.

How Draw :



1. 이벤트를 표기한다. (세세한 프로세스)  
2. 시스템의 상태를 노드로 표시하고 이를 노드로 연결한다.  
3. 어떤 사건이 발생할 때, 시스템은 하나의 상태에서 다른 상태로 이동함을 보여주는 다이어그램을 만들면 된다.

**8. (a) What is Design Thinking? How can it help your class project?**

디자인적 사고를 기반으로 사람 중심의 공감을 통해 새롭게 문제점을 해석하고 풀어내어 창의적인 혁신을 촉진하는 사고방식이다.

**(b)   Explain the Design Thinking Process with an example (e.g., in your class project)**

Design Thinking Process : Empathize – Define – Ideate – Prototype – Test

Empathize – 고객의 불편함이나 문제 상황을 고객의 입장에서 공감하는 것이다.

Ex. 결혼식 등의 특별한 날이나, 여행지를 방문했을 때 추억을 남기기 위해 사진을 찍는다. 하지만 팀원 중 누군가 사진을 찍기 위해 빠지는 것이 안타깝고, 지나가는 행인에게 부탁하기에는 미덥지 못하다. 지금 이 순간은 한 번 지나가면 다시 오지 않기 때문이다.

Define – 문제를 정의하는 것이다

Ex. 어떻게 하면 특별한 순간을 만족스러운 사진으로 담아줄 수 있는 사람을 편하게 찾을 수 있을까?

Ideate – 문제에 적합한 해결방안을 위해 아이디어를 창출하는 것이다.

Ex. 사진을 찍는 능력에 대한 평가를 수행할 수 있고, 사진을 잘 찍는 사람(Ex. 아마추어 사진작가, 취미가 사진촬영인 일반인 등)과 사진 촬영을 의뢰하고 싶은 사람을 매칭해주자.

Prototype – 빠르게 프로토타입을 제작하는 것이다.

Ex. 사진을 잘 찍는 사람들과 사진 촬영을 의뢰하고 싶은 사람들을 매칭해주는 플랫폼을 제작한다.

Test – 테스트하고 반복하라.

Ex. 플랫폼을 제작하여 런칭한 뒤 사용자들의 반응을 보고 피드백을 받아 기능들에 대한 수정을 진행하거나 새로운 방식의 해결방안을 만들어본다

**(c)   What are three tools of How to Empathize? What tool did your project use? What are their results? Good or bad? How to improve it better in next time?**

세 가지 방법은 관찰하기, 인터뷰하기, 직접 경험하기이다.

우리는 세 방법 모두를 통해 empathize를 수행했다. 관찰하기의 경우 표면적인 모습들 밖에 볼 수 없어 특별한 결과를 얻지 못 했다. 인터뷰하기의 경우 졸업사진 촬영에 대한 이야기를 들을 수 있었고, 직접 경험하기는 팀원 각자의 여행 경험에서 그 불편함을 느낄 수 있었다. 전반적으로 고객의 불편함을 체험하는 데에는 충분했으나, 아쉬웠던 점은 더 많은 사람들에 대한 인터뷰를 수행해 어느 정도의 비율이 그 문제를 불편하다고 느끼는지 파악하지 못했던 것이다. 따라서 다음에는 더 많은 수의 인터뷰를 통해 그 정도를 파악해볼 수 있을 것이다.

**(d)   What is “Empathy Map” (e.g., think, say, feel, do)? Explain it with an exmaple in your class project**

공감지도(Empathy Map)은 고객들의 불편함에 대해 관찰, 인터뷰 한 결과들을 통해 취합한 정보를 통합하며 깊은 공감을 할 수 있게 도움을 주는 수단이다.

SAY : 고객들이 무엇을 말 하였는가. Ex. 여행을 갔다 왔는데 거기서 지나가는 행인이 찍어준 사진이 마음에 안 들어요. 여자친구와의 기념일을 맞아 특별한 데이트를 하는데, 전문 사진사를 고용하기는 비싸고 셀카로 사진을 찍고 싶지는 않아요.

THINK : 고객들은 어떤 생각을 했는가. 중요하게 생각하는 가치나 신념은 무엇인가. Ex. 셀카로 사진을 찍을 수는 있지만, 다른 사람이 배경과 함께 찍어준 사진이 가지고 싶다. 단체 사진을 찍을 때 누군가 사진을 찍어주기 위해 빠지는 것이 싫다. 여행을 가서 사진을 남기는데 많은 시간을 들이는 것이 아깝다.

DO : 고객들은 어떤 행동을 했는가. Ex. 여행지에 가면 사진촬영을 부탁해서 나온 결과물이 마음에 들지 않아 다시 다른 사람에게 촬영을 부탁하는 경우가 종종 있다.

FEEL : 고객은 어떤 감정을 느꼈는가. Ex. 친구와 단 둘이 여행을 갔는데, 서로 찍어준 사진은 많지만 함께 찍은 사진이 많지 않은 데에 대한 아쉬움. 지나가는 행인들 중 누가 사진을 잘 찍는지 모르는 불확실성에 대한 걱정. 사진을 부탁하는 대상의 신뢰도에 대한 불안.

**(e)   Describe POV (Point-of-View) of your class project.**

특별한 순간을 사진으로 남기고 싶어하는 사람은

만족스러운 사진을 찍어줄 신뢰할 만한 사진작가를 구하기가 어려우므로

믿을 만한 사진작가를 저렴하고 빠르게 매칭해 줄 수 있는 플랫폼이 필요하다.