REPORT

과목명 | 소프트웨어공학

담당교수 | 김명호

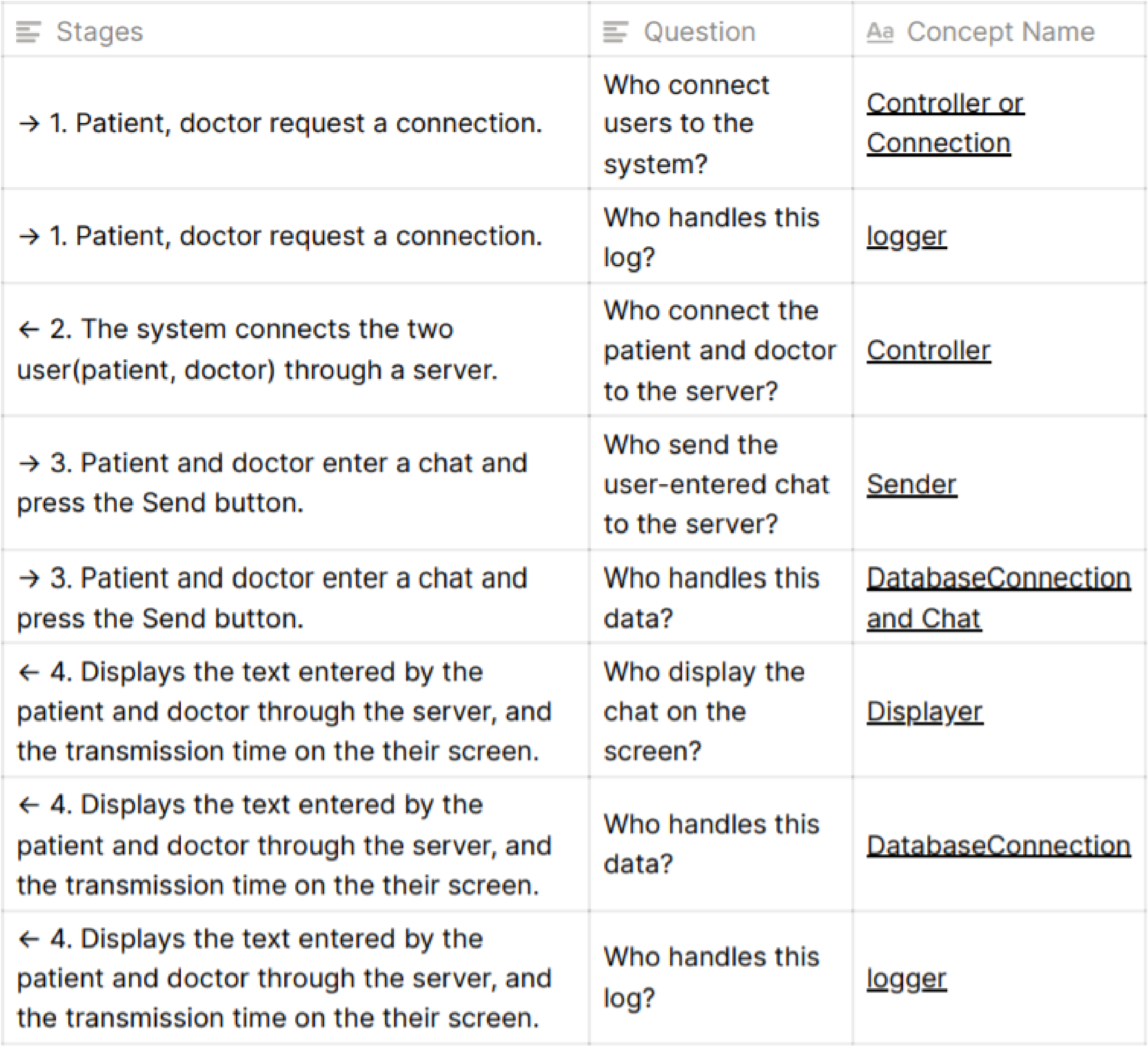
학과 | 소프트웨어학과

학번	이름
20194461	신혜진
20194538	이나혁
20193418	이하윤
20192939	이하은
20193574	정설희
20193878	최은서

-Table of Content

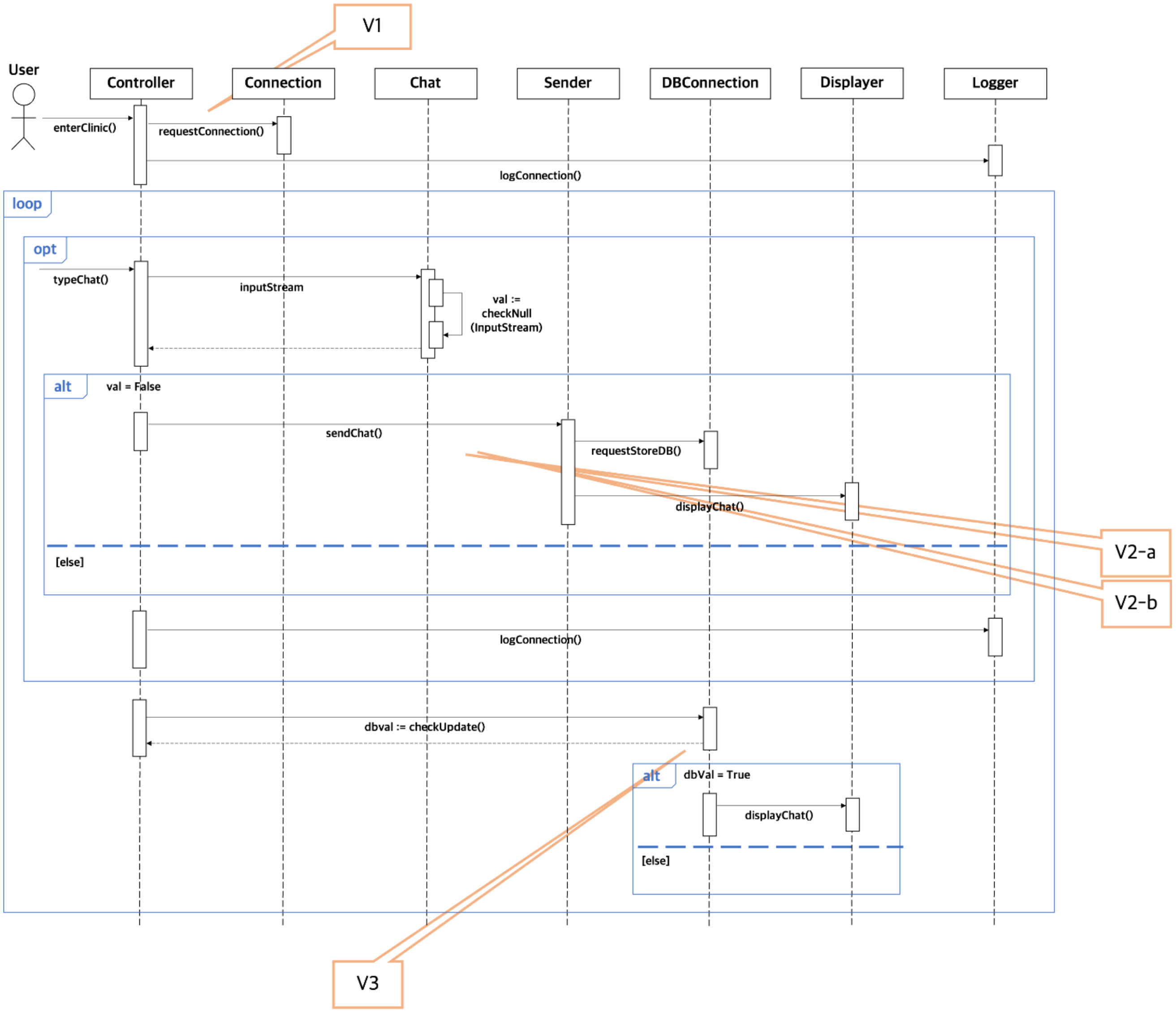
1. **UC-1**

Scenario Walkthrough



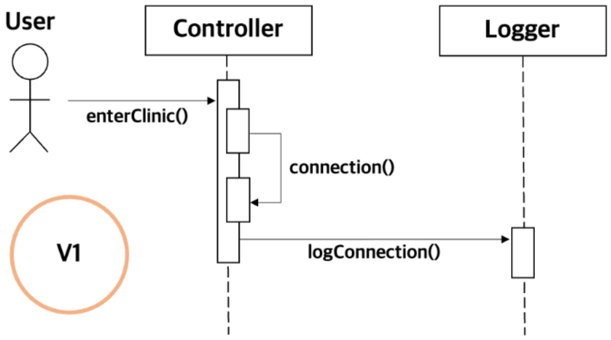
≡ Stages	≡ Question	Aa Concept Name
→ 1. Patient, doctor request a connection.	Who connect users to the system?	Controller or Connection
→ 1. Patient, doctor request a connection.	Who handles this log?	logger
← 2. The system connects the two user(patient, doctor) through a server.	Who connect the patient and doctor to the server?	Controller
→ 3. Patient and doctor enter a chat and press the Send button.	Who send the user-entered chat to the server?	Sender
→ 3. Patient and doctor enter a chat and press the Send button.	Who handles this data?	DatabaseConnection and Chat
← 4. Displays the text entered by the patient and doctor through the server, and the transmission time on the their screen.	Who display the chat on the screen?	Displayer
← 4. Displays the text entered by the patient and doctor through the server, and the transmission time on the their screen.	Who handles this data?	DatabaseConnection
← 4. Displays the text entered by the patient and doctor through the server, and the transmission time on the their screen.	Who handles this log?	logger

Sequence Diagram



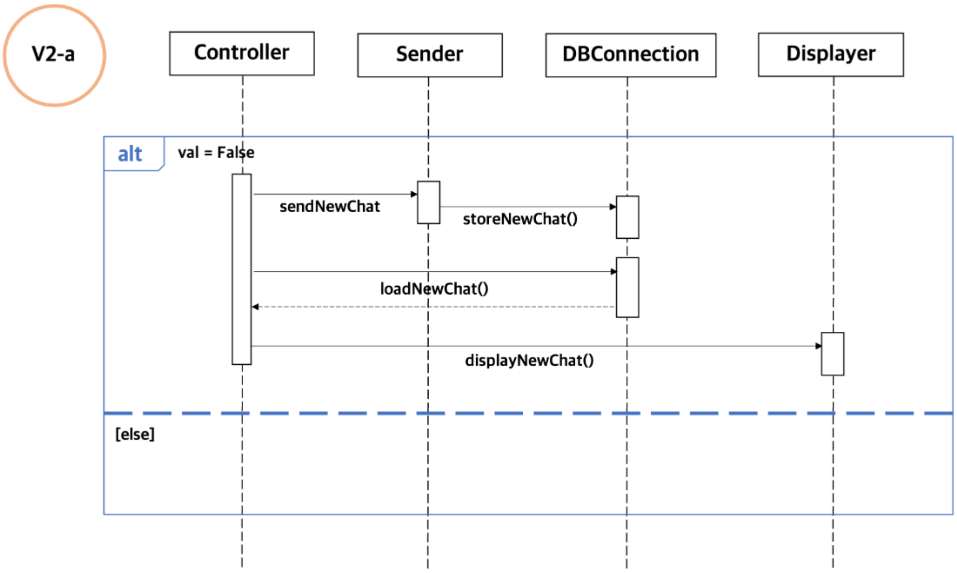
- **Variation 1**

- Instead of requesting connection to the Connection as a return value for the user's connection request, the controller directly connects the user.
- Advantage: Less coupling(Contorller - Connection), Less Class needed
- Disadvantage: Extra responsibility for Controller (low cohesion)



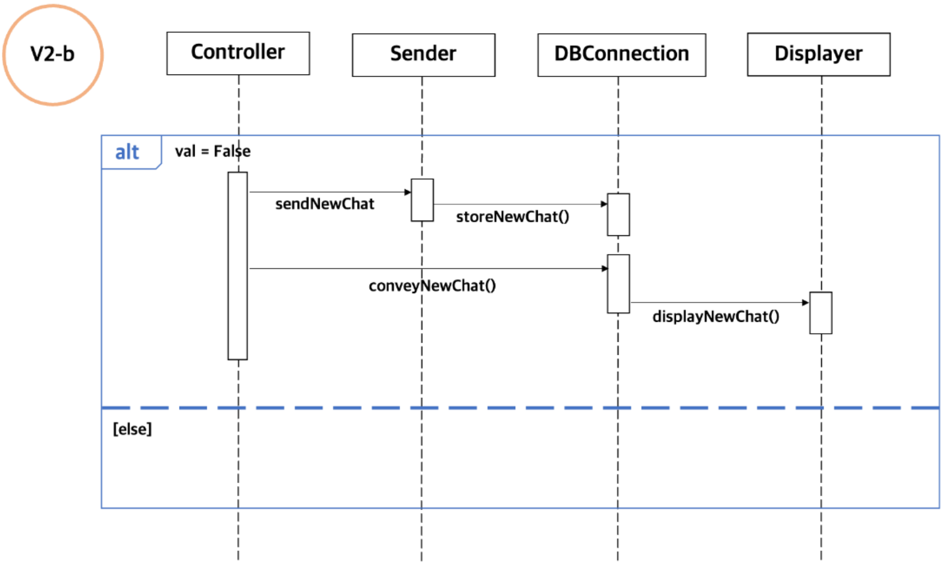
- **Variation 2-a**

- Instead of returning a request from the Sender to the DBConnection and the display, respectively, for a request to send a chat, the controller plays the role.
- Advantage: Clear role(Sender)
- Disadvantage: Longer communication chain, Extra responsibility for Controller(low cohesion)



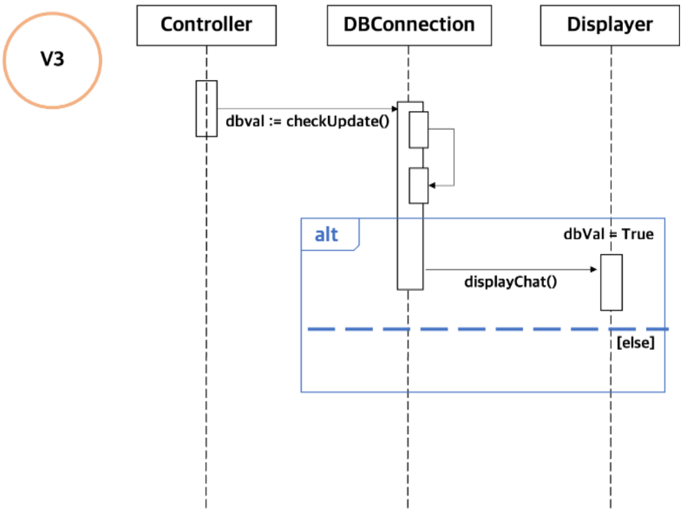
- **Variation 2-b**

- Instead of the DBConnector returning Chat data, it passes the Chat to the Displayer to request display.
- Advantage: Shorter communication chain, less coupling(DBConnector - Controller)
- Disadvantage: Extra responsibility for DBConnector(low cohesion)

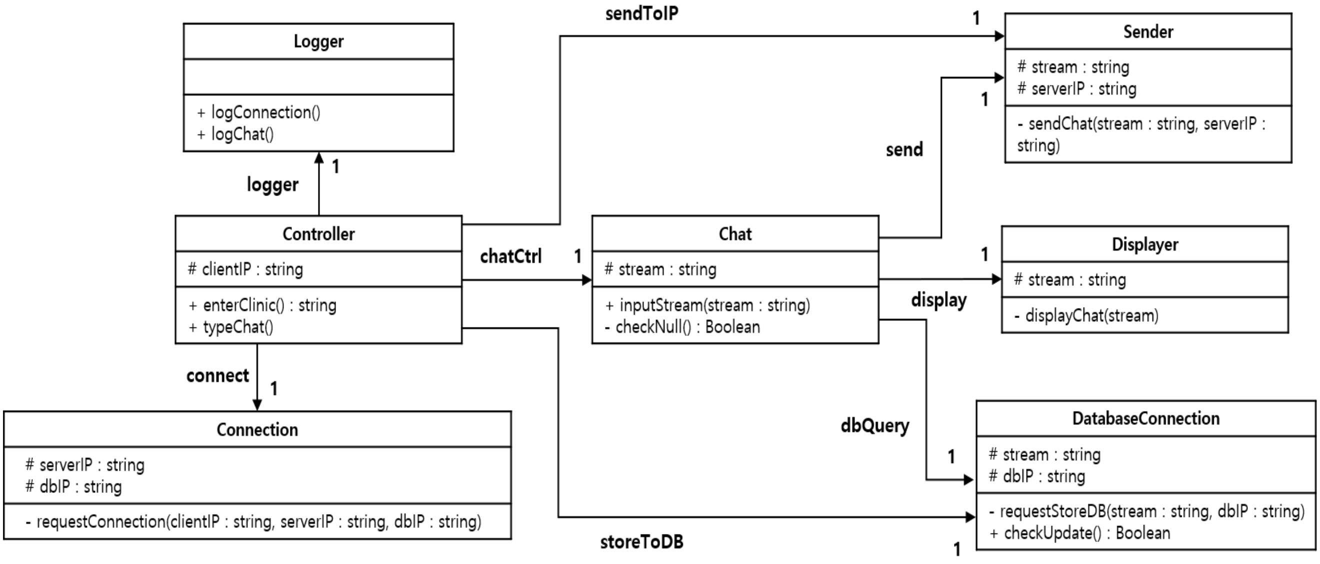


- **Variation 3**

- Instead of the DBConnector returning the dbval to the controller, it determines the variable in the DBConnector and requests it directly to the Displayer.
- Advantage: Shorter communication chain
- Disadvantage: Extra responsibility for DBConnector(low cohesion)



Class Diagram

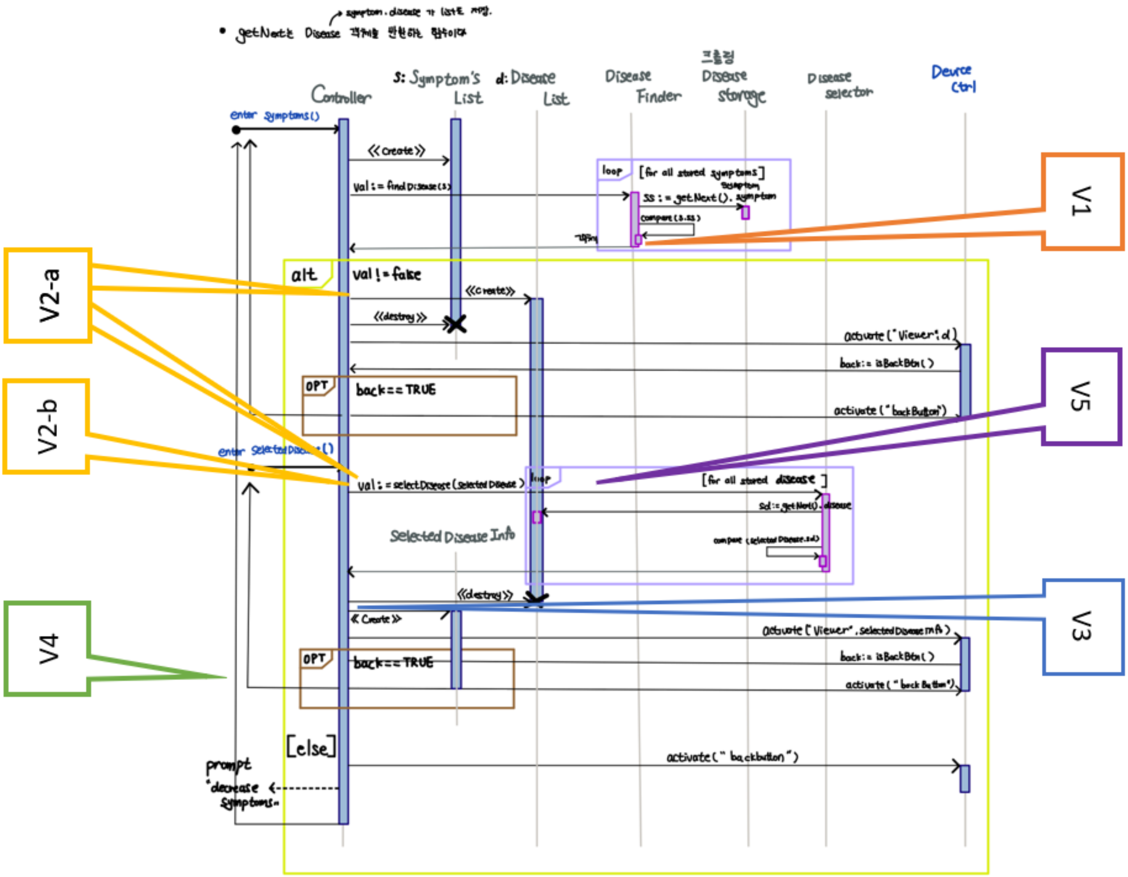


2. UC-4

Scenario Walkthrough

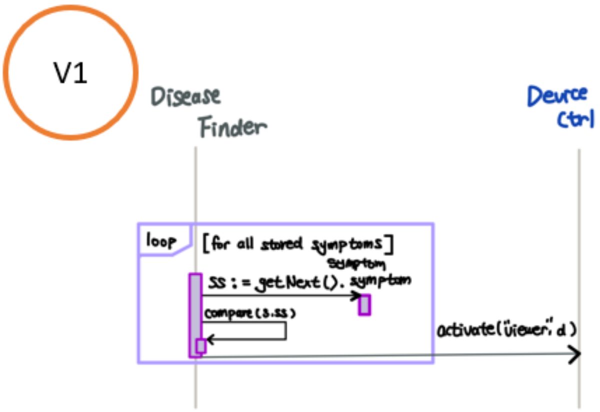
Stages	Question	Aa Concept Name
←1. Initialize the value of the symptoms to find a list of diseases	Who initialize the value of symptoms?	Controller
←2. Show the list of disease	Who perform to find the disease?	Controller or Disease finder
←3. Citizen selects one of the disease in the disease list	Who handles this data?	Controller
←4. Citizen can view the disease information	Who gives disease information to viewer?	Controller or Disease List
←5. Add 진료과 to citizen's DB	Who handles this data?	Controller or Disease List

Sequence Diagram



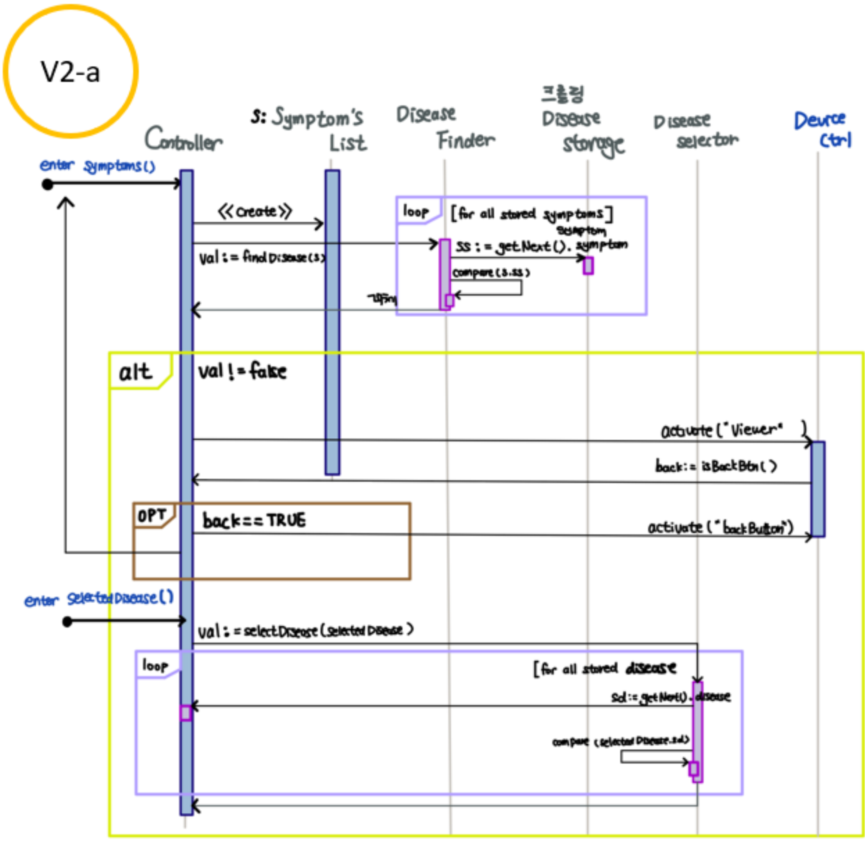
- **Variation 1**

- Instead of original solution where the controller activates the viewer with the return value of disease finder, diseases finder activates viewer directly with related diseases
- Advantage: Shorter communication chain, less coupling(disease finder - disease list)
- Disadvantage: Extra responsibility for disease finder (low cohesion)



- **Variation 2-a**

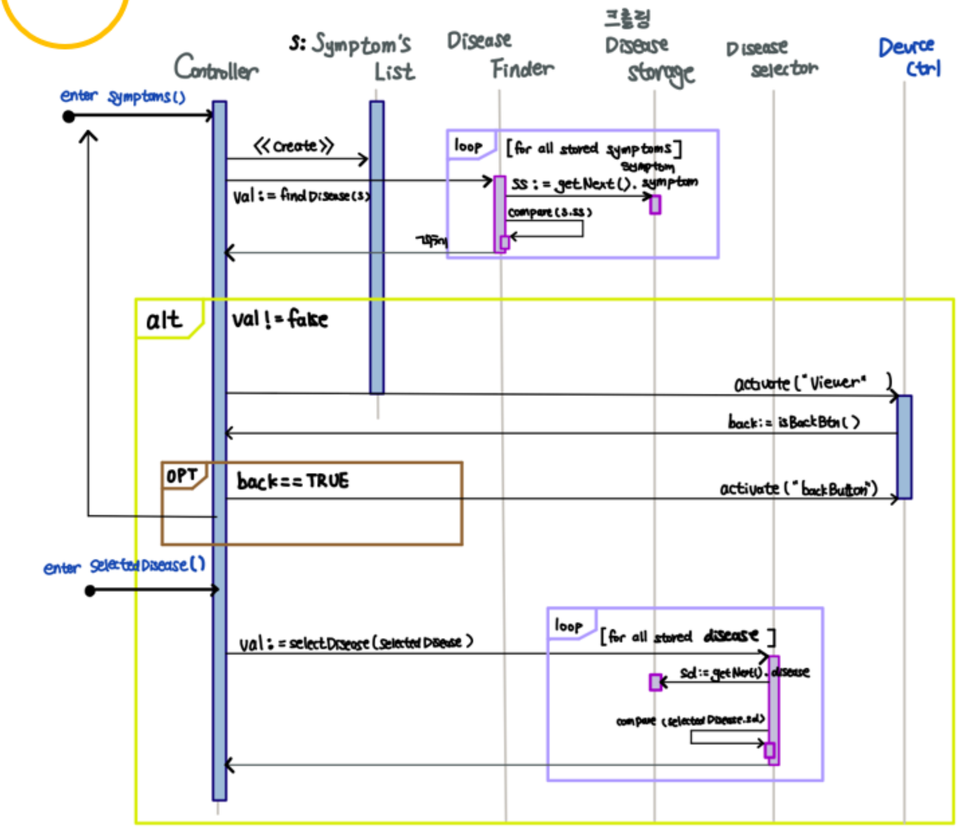
- Instead of original solution where the symptoms-related diseases are stored in disease list, it stores diseases in controller
- Advantage: Less coupling(disease finder - disease list)
- Disadvantage: Extra storage space required for controller, controller should not be involved in device-control details (low cohesion, low specialty)



- **Variation 2-b**

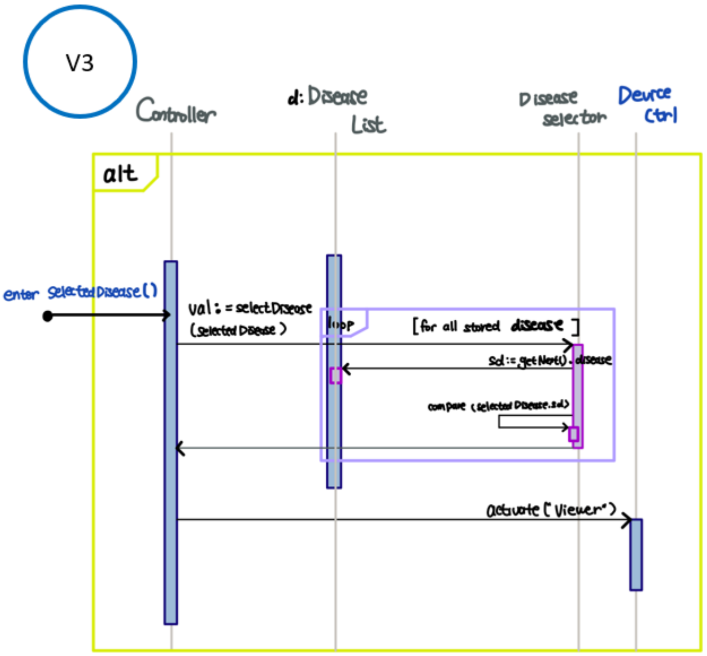
- Instead of original solution where the symptoms-related diseases are stored in disease list, it doesn't store related disease
- Advantage: No extra storage needed, less coupling(disease finder - disease list)
- Disadvantage: When user selects the disease, disease selector should find its information in disease storage instead of disease list and it is not efficient

V2-b



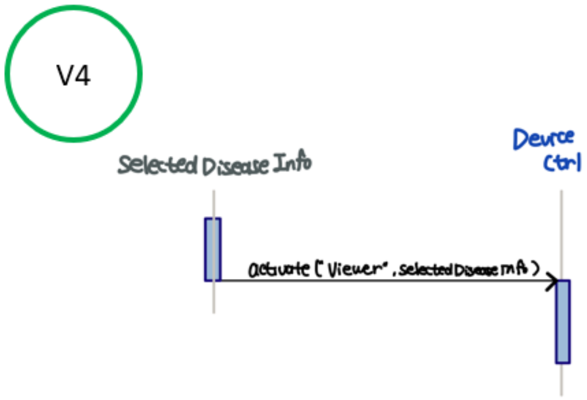
- **Variation 3**

- **Instead of original solution where the user-selected diseases are stored in selected disease info, it stores diseases in controller**
- **Advantage: Less coupling(disease selector - selected disease info)**
- **Disadvantage: Extra storage space required for controller, controller should not be involved in device-control details (low cohesion)**



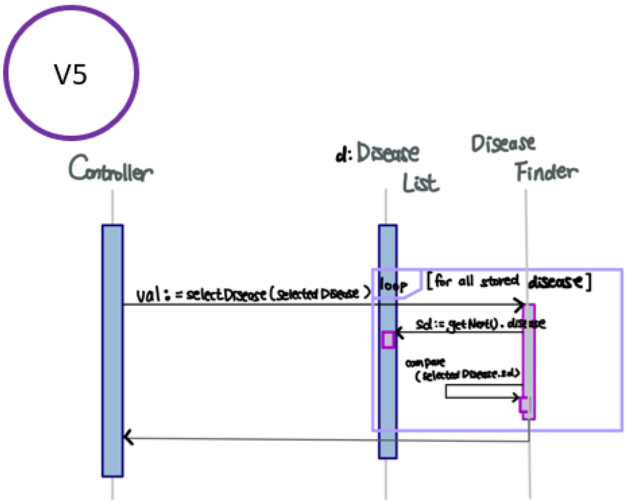
- **Variation 4**

- Instead of original solution where the controller activates the viewer with the information of selected disease info, diseases selector activates viewer directly
- Advantage: Shorter communication chain, less coupling(disease selector - selected disease info)
- Disadvantage: Extra responsibility for disease selector (low cohesion, low specialty), selected disease info have to be stored anyway

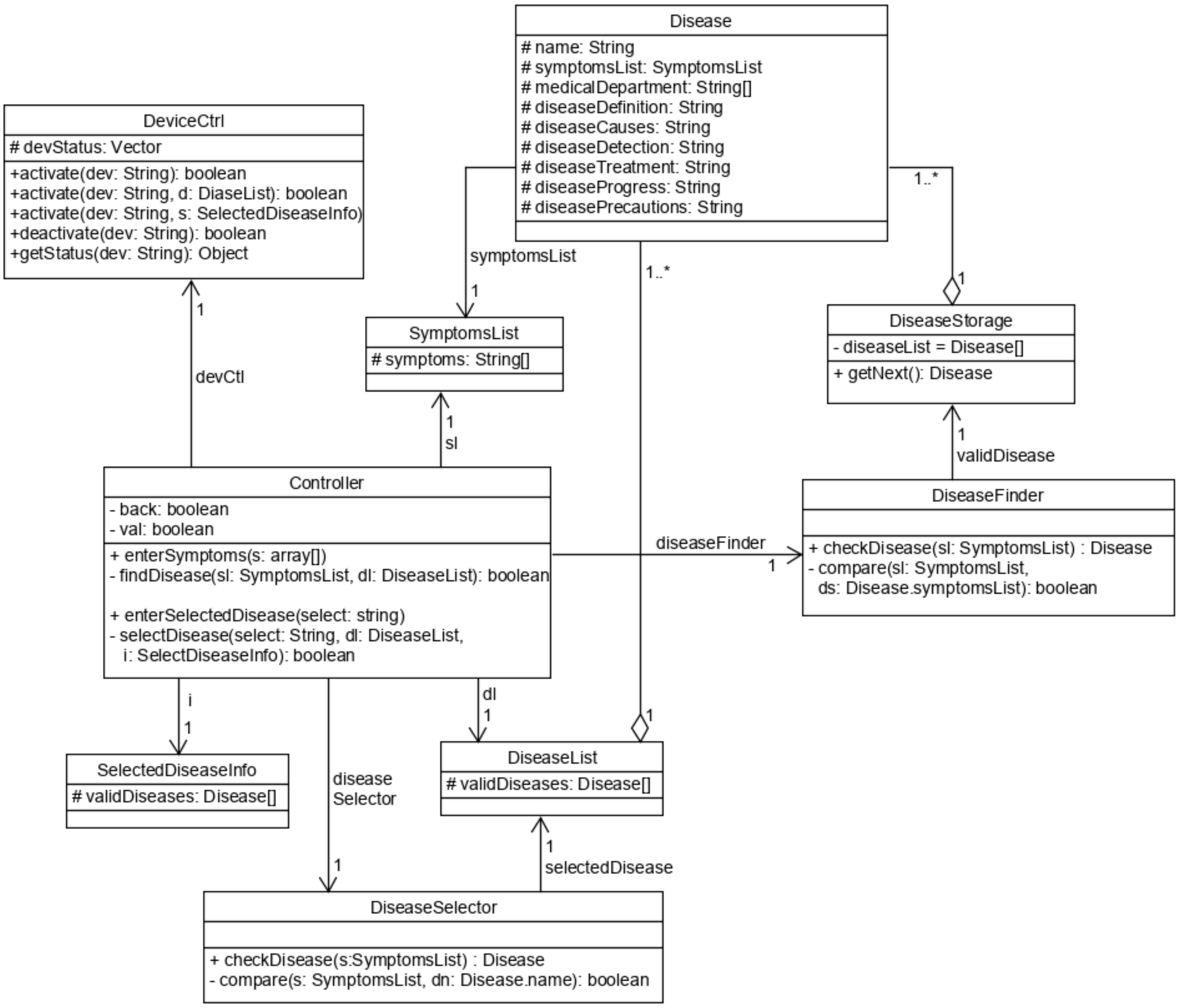


- **Variation 5**

- Instead of original solution where the disease selector searches diseases, disease finder does that job.
- Advantage: Less class needed
- Disadvantage: Extra responsibility for disease finder (low cohesion, low specialty)

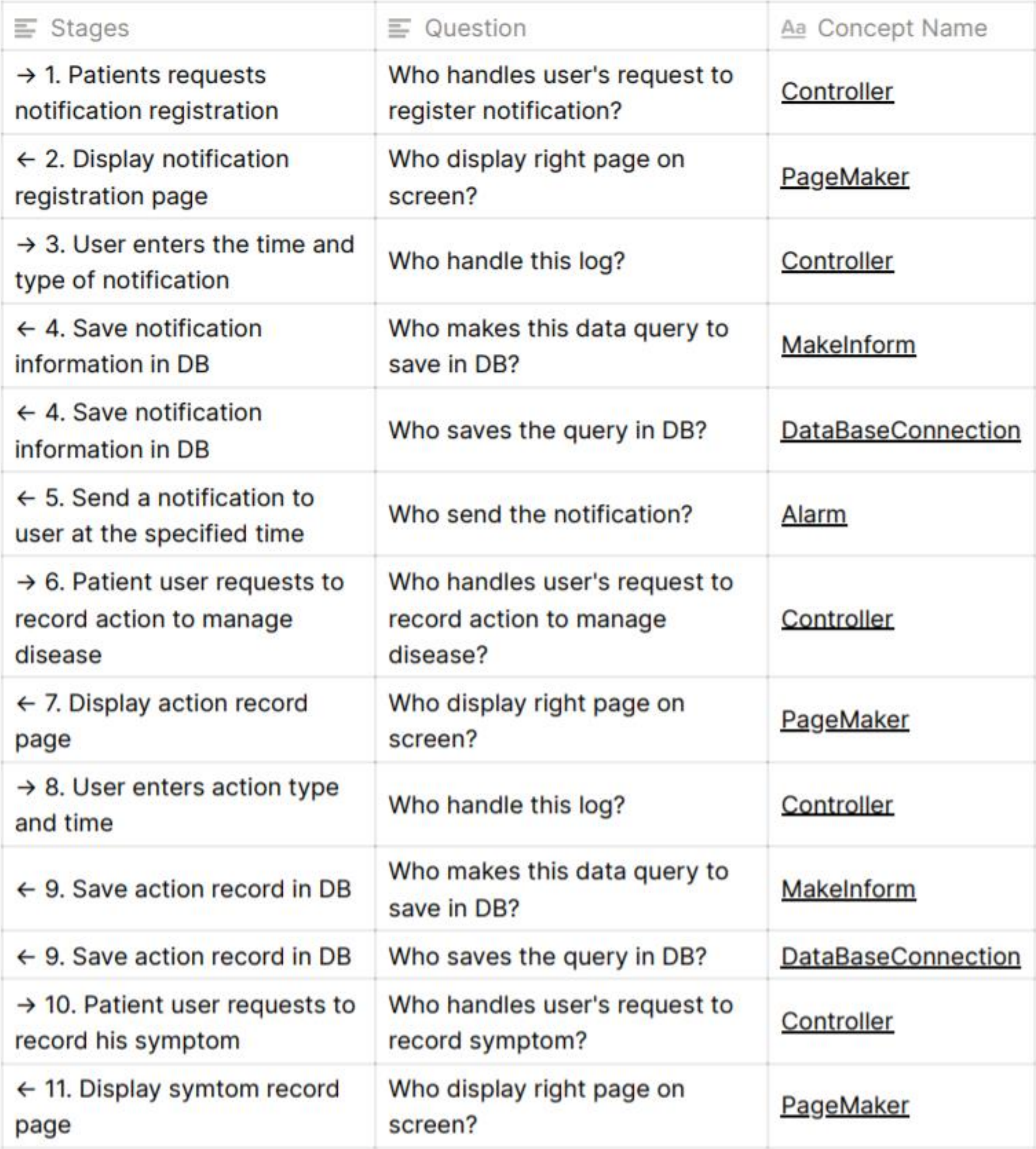
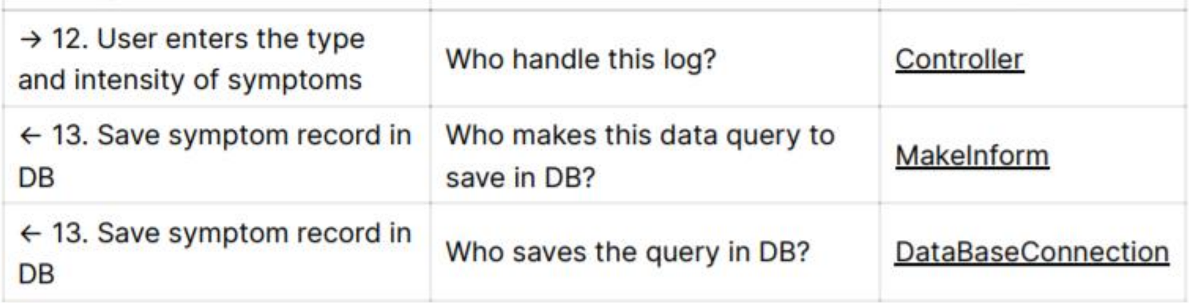


Class Diagram


Disease
# name: String
# symptomsList: SymptomsList
# medicalDepartment: String[]
# diseaseDefinition: String
# diseaseCauses: String
# diseaseDetection: String
# diseaseTreatment: String
# diseaseProgress: String
# diseasePrecautions: String
DeviceCtrl
# devStatus: Vector
+activate(dev: String): boolean
+activate(dev: String, d: DiaseList): boolean
+activate(dev: String, s: SelectedDiseaseInfo)
+deactivate(dev: String): boolean
+getStatus(dev: String): Object
symptomsList
1
1..*
1..*
1
SymptomsList
# symptoms: String[]
DiseaseStorage
- diseaseList = Disease[]
+ getNext(): Disease
1
devCtl
1
sl
1
validDisease
Controller
- back: boolean
- val: boolean
+ enterSymptoms(s: array[])
- findDisease(sl: SymptomsList, dl: DiseaseList): boolean
+ enterSelectedDisease(select: string)
- selectDisease(select: String, dl: DiseaseList,
i: SelectDiseaseInfo): boolean
diseaseFinder
1
DiseaseFinder
+ checkDisease(sl: SymptomsList) : Disease
- compare(sl: SymptomsList,
ds: Disease.symptomsList): boolean
i
1
dl
1
1
SelectedDiseaseInfo
# validDiseases: Disease[]
disease
Selector
DiseaseList
# validDiseases: Disease[]
1
selectedDisease
1
DiseaseSelector
+ checkDisease(s:SymptomsList) : Disease
- compare(s: SymptomsList, dn: Disease.name): boolean

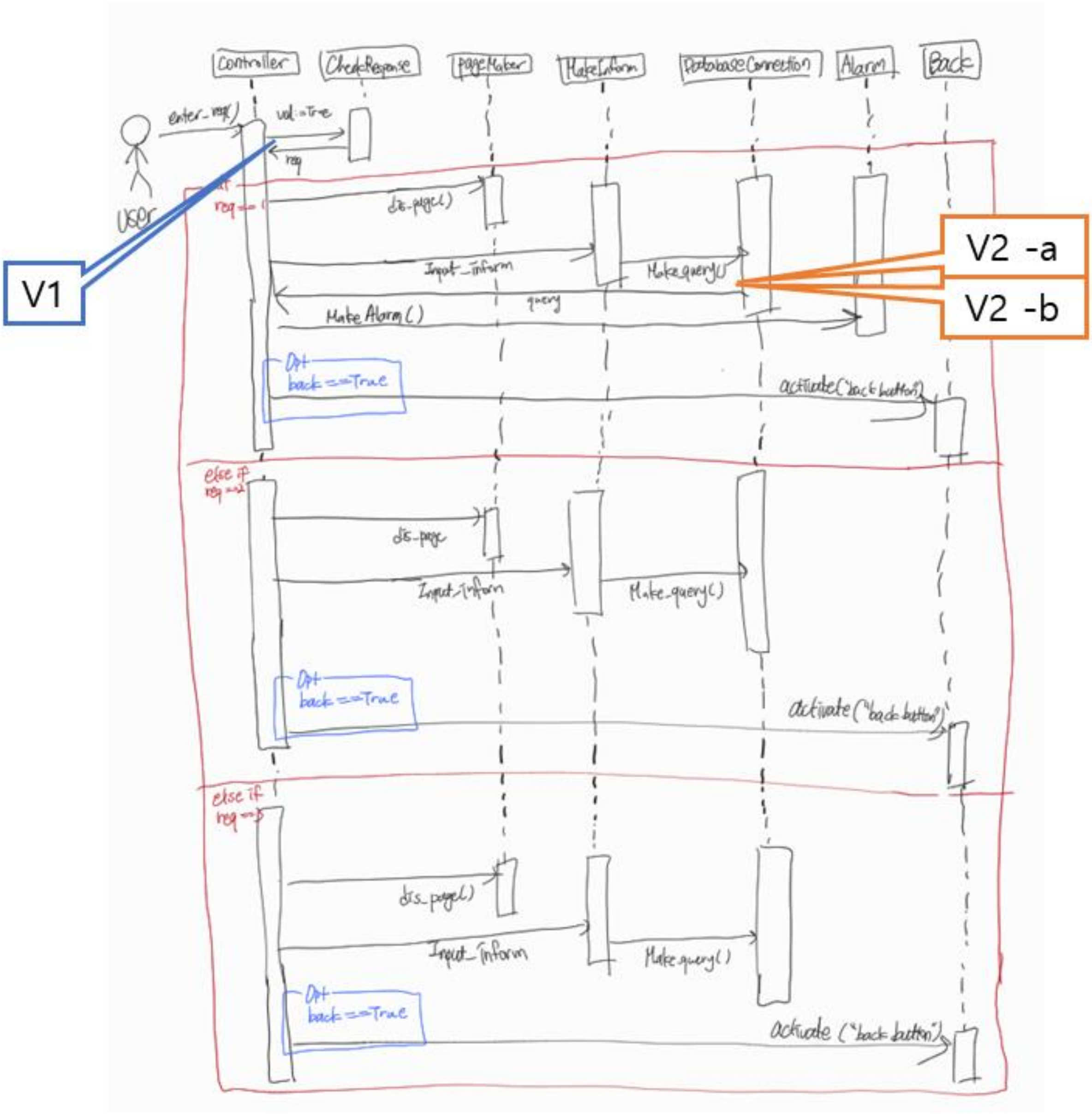
3. UC-10

Use Cases

Stages	Question	Concept Name
→ 1. Patients requests notification registration	Who handles user's request to register notification?	Controller
← 2. Display notification registration page	Who display right page on screen?	PageMaker
→ 3. User enters the time and type of notification	Who handle this log?	Controller
← 4. Save notification information in DB	Who makes this data query to save in DB?	MakeInform
← 4. Save notification information in DB	Who saves the query in DB?	DataBaseConnection
← 5. Send a notification to user at the specified time	Who send the notification?	Alarm
→ 6. Patient user requests to record action to manage disease	Who handles user's request to record action to manage disease?	Controller
← 7. Display action record page	Who display right page on screen?	PageMaker
→ 8. User enters action type and time	Who handle this log?	Controller
← 9. Save action record in DB	Who makes this data query to save in DB?	MakeInform
← 9. Save action record in DB	Who saves the query in DB?	DataBaseConnection
→ 10. Patient user requests to record his symptom	Who handles user's request to record symptom?	Controller
← 11. Display symtom record page	Who display right page on screen?	PageMaker
→ 12. User enters the type and intensity of symptoms	Who handle this log?	Controller
← 13. Save symptom record in DB	Who makes this data query to save in DB?	MakeInform
← 13. Save symptom record in DB	Who saves the query in DB?	DataBaseConnection

Sequence Diagram

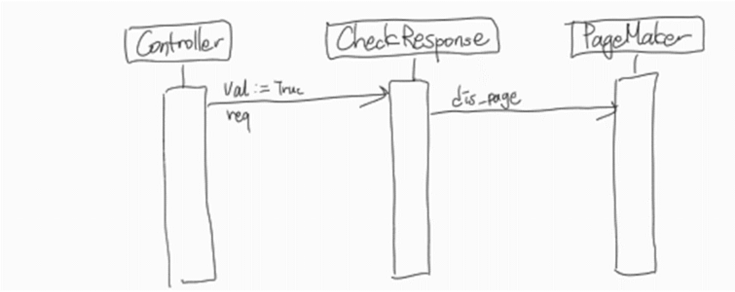


- **Variation 1**

Instead of CheckResponse sends a req value to Controller and Controller sends a signal to PageMaker depending on the req value, CheckResponse sends a signal directly to PageMaker.

- Advantage : shorter communication chain
- Disadvantage : Extra respoonsibility for CheckResponse

V1

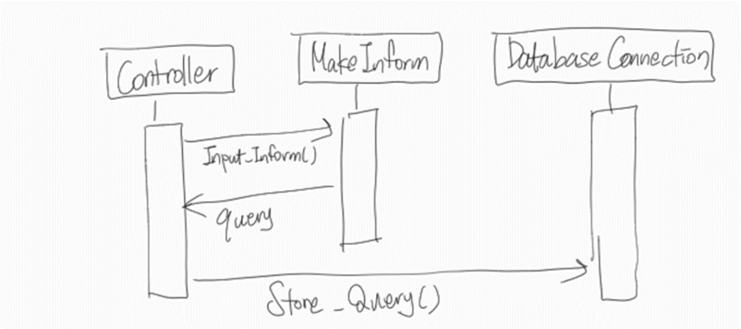


- **Variation 2-a**

Instead of store query directly from MakeInform to DatabaseConnection, save the query via the Controller.

- Advantage : shorter MakeAlarm chain (Controller knows the query)
- Disadvantage : longer query save chain

V2 -a

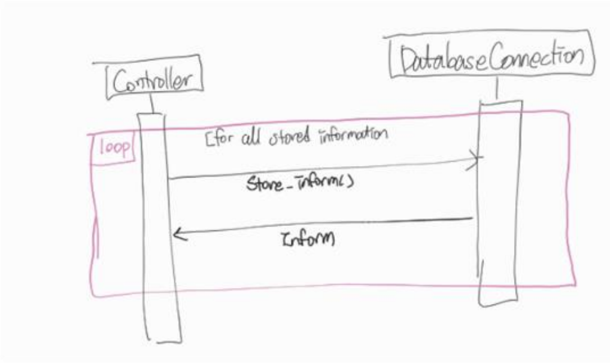


- **Variation 2-b**

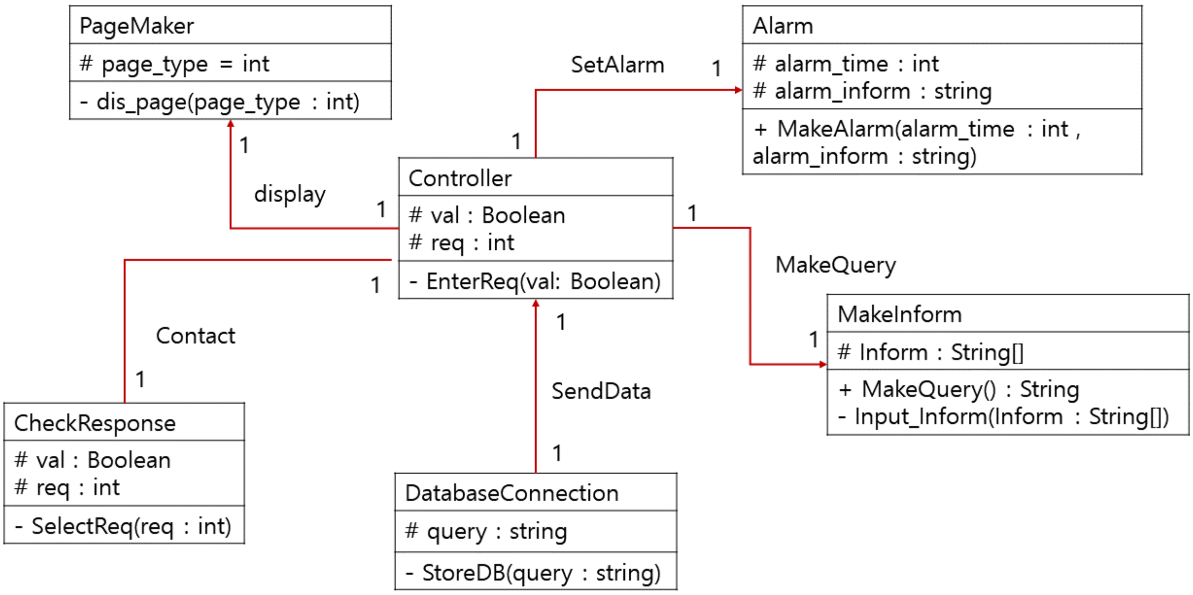
Instead of converting multiple information into a single query, each information is stored separately.

- **Advantage : can skip MakeQuery course**
- **Disdvantage : send multiple data when referencing data from DB (loop occurs)**

V2 -b



Class Diagram


PageMaker
# page_type = int
- dis_page(page_type : int)
Alarm
# alarm_time : int
# alarm_inform : string
+ MakeAlarm(alarm_time : int , alarm_inform : string)
SetAlarm
1
1
1
display
1
Controller
# val : Boolean
# req : int
- EnterReq(val: Boolean)
1
MakeQuery
1
1
Contact
1
1
SendData
1
MakeInform
# Inform : String[]
+ MakeQuery() : String
- Input_Inform(Inform : String[])
CheckResponse
# val : Boolean
# req : int
- SelectReq(req : int)
DatabaseConnection
# query : string
- StoreDB(query : string)