

프론트엔드 서버 설치 및 구동 매뉴얼

(주) 젠토

* 이 문서는 다음 환경을 기반으로 작성되었습니다.*

* OS : Ubuntu 24.04.3 LTS

* Python : 3.12.3

* 매뉴얼은 Python 관련해서 python 과 pip 가, lisp 관련해서 sbcl 과 lisp 관련 패키지가 모두 설치된 환경에서 진행되었음

실행명령 요약

```
# 명령 수행 폴더는 frontend 소스 폴더
```

ETRI 모듈 구동

```
$ sbcl --load "quicklisp/setup.lisp" --load "wde/usability/extend/load-act-r.lisp" --load  
"wde/usability/extend/usability/system_interface.lisp"
```

frontend 서버 시작

```
$ python3 -m venv .venv  
$ source .venv/bin/activate  
$ pip3 install -r requirements.txt  
$ python3 main.py
```

1. 소스 구조 및 기본 환경 설정

제공되는 frontend 서버의 소스 구조는 다음과 같다.

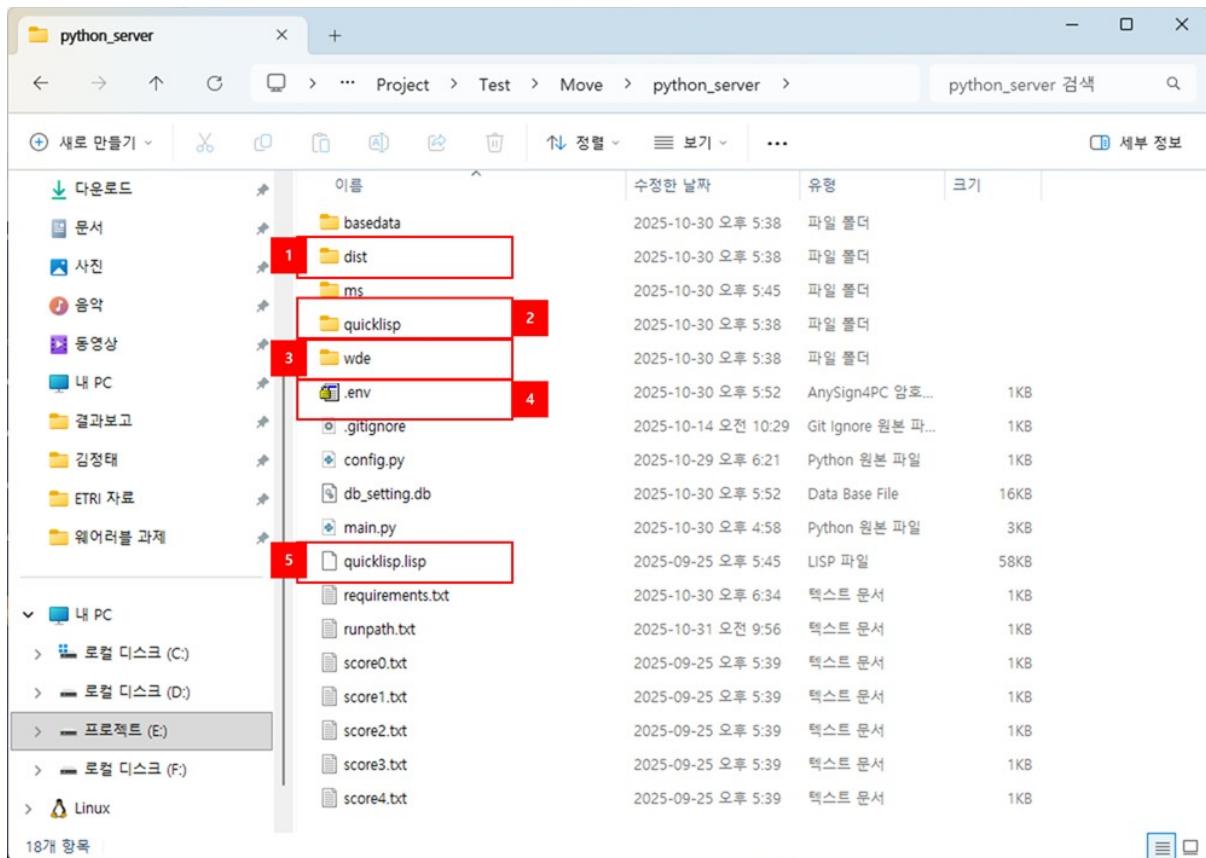


그림 1 소스 구조

번호	설명
1	UI 화면 소스 폴더
2	quicklisp 다운로드 폴더
3	ETRI 제공 분석용 소스 파일 폴더
4	서버 환경 설정 파일
5	quicklist 실행 파일

제공된 소스에는 분석을 위한 quicklisp 관련 파일 및 actr7 관련 파일이 모두 포함된 상태이며, ETRI에서 제공한 분석용 코드를 실행할 수 있는 상태로 제공되었다.

현재 ETRI에서 제공된 실행 모듈은 wde 폴더에 각 기능별로 해당 폴더에 나누어 저장되어 있으며 구동을 위한 참조 파일들은 해당 폴더의 extend 폴더 아래에 배치되어 있다.

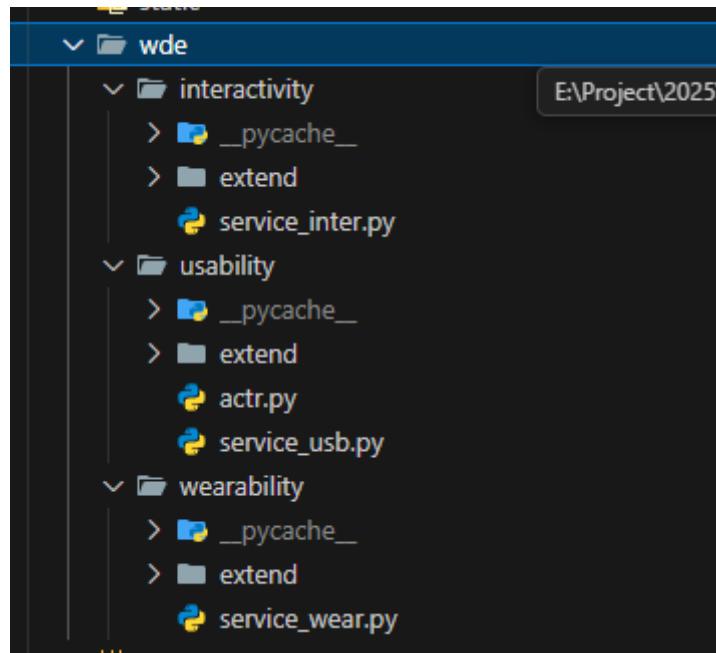


그림 2 ETRI 제공 모듈 위치

만일 sbcl 을 통한 실행명령을 입력했을 때 ETRI 제공 모듈이 동작하지 않는다면 quicklisp 와 관련된 사항을 다시 다운받아서 사용하기 바람.

이제 Python 서버 구동을 위한 환경 설정에 대해 기술하겠다.

먼저 서버 구동을 위한 가상환경을 설정하겠다.

필요없으면 이 과정을 생략해도 된다.

Frontend 소스 코드가 있는 폴더로 이동한다.

다음 명령을 통해 가상환경 폴더를 생성한다.

```
$ python3 -m venv .venv
```

```
zento@zento-SEi:~/Test/Server/python_server$ python3 -m venv .venv
zento@zento-SEi:~/Test/Server/python_server$ ls -a
.          db_setting.db  main.py      quicklisp.lisp    score1.txt  .venv
..         dist            ms          requirements.txt  score2.txt  wde
basedata   .env           __pycache__ runpath.txt    score3.txt
config.py  .gitignore     quicklisp    score0.txt      score4.txt
zento@zento-SEi:~/Test/Server/python_server$
```

그림 3 가상환경 생성 결과

해당 폴더가 생성되었다면 가상환경을 사용할 준비가 된 것이다.

가상환경을 활성화하겠다.

```
$ source .venv/bin/activate
```

```
zento@zento-SEi:~/Test/Server/python_server$ python3 -m venv .venv
zento@zento-SEi:~/Test/Server/python_server$ ls -a
.          db_setting.db  main.py      quicklisp.lisp    score1.txt  .venv
..         dist            ms          requirements.txt  score2.txt  wde
basedata   .env           __pycache__ runpath.txt    score3.txt
config.py  .gitignore     quicklisp    score0.txt      score4.txt
zento@zento-SEi:~/Test/Server/python_server$ source .venv/bin/activate
(.venv) zento@zento-SEi:~/Test/Server/python_server$
```

그림 4 가상환경 활성화 결과

가상환경이 활성화되면 해당 터미널을 통해 필요한 패키지를 설치하겠다.

패키지 내역은 requirements.txt에 정의되어 있으며 다음 명령을 입력하여 설치하겠다.

```
$ pip3 install -r requirements.txt
```

The image shows two terminal windows side-by-side. The top window displays the directory structure of a Python project within a virtual environment, including files like db_setting.db, main.py, quicklisp.lisp, score1.txt, .env, dist, ms, requirements.txt, score2.txt, runpath.txt, score3.txt, config.py, .gitignore, quicklisp, score0.txt, score4.txt, and wde. The bottom window shows the output of the pip3 install command, which lists numerous packages being downloaded and installed from cached sources, including six, typing_extensions, tzdata, watchdog, werkzeug, pytz, and many others, totaling 35.7 MB.

```
zento@zento-SEi:~/Test/Server/python_server$ python3 -m venv .venv
zento@zento-SEi:~/Test/Server/python_server$ ls -a
.
..
db_setting.db  main.py      quicklisp.lisp    score1.txt  .venv
dist           ms          requirements.txt  score2.txt  wde
basedata       .env        __pycache__     runpath.txt score3.txt
config.py      .gitignore   quicklisp      score0.txt  score4.txt
zento@zento-SEi:~/Test/Server/python_server$ source .venv/bin/activate
(.venv) zento@zento-SEi:~/Test/Server/python_server$ pip3 install -r requirements.txt
```



```
zento@zento-SEi:~/Test/Server/python_server$ pip3 install -r requirements.txt
1 (35.7 MB)
Using cached six-1.17.0-py2.py3-none-any.whl (11 kB)
Using cached typing_extensions-4.15.0-py3-none-any.whl (44 kB)
Using cached tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Using cached watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl (79 kB)
Using cached werkzeug-3.1.3-py3-none-any.whl (224 kB)
Installing collected packages: pytz, aniso8601, watchdog, tzdata, typing_extensions, six, rpds-py, python-dotenv, pyparsing, pillow, packaging, numpy, MarkupSafe, kiwisolver, itsdangerous, importlib_resources, fonttools, cycler, colorama, click, blinker, attrs, Werkzeug, scipy, referencing, python-dateutil, Jinja2, dotenv, contourpy, pandas, matplotlib, jsonschema-specifications, Flask, jsonschema, flask-cors, flask-restx
Successfully installed Flask-3.1.2 Jinja2-3.1.6 MarkupSafe-3.0.3 Werkzeug-3.1.3 aniso8601-10.0.1 attrs-25.4.0 blinker-1.9.0 click-8.3.0 colorama-0.4.6 contourpy-1.3.3 cycler-0.12.1 dotenv-0.9.9 flask-cors-6.0.1 flask-restx-1.3.2 fonttools-4.60.1 importlib_resources-6.5.2 itsdangerous-2.2.0 jsonschema-4.25.1 jsonschema-specifications-2025.9.1 kiwisolver-1.4.9 matplotlib-3.10.7 numpy-2.3.4 packaging-25.0 pandas-2.3.3 pillow-12.0.0 pyparsing-3.2.5 python-dateutil-2.9.0.post0 python-dotenv-1.2.1 pytz-2025.2 referencing-0.37.0 rpds-py-0.28.0 scipy-1.16.3 six-1.17.0 typing_extensions-4.15.0 tzdata-2025.2 watchdog-6.0.0
(.venv) zento@zento-SEi:~/Test/Server/python_server$
```

그림 5 패키지 설치 결과

가상환경에 패키지 설치가 완료되었다.

이제 서버를 구동할 수 있는 환경이 모두 갖추어졌다.

다음장에서 서버 구동에 대한 절차를 이어가겠다.

2. 서버 실행

서버 실행을 위해 먼저 ETRI 제공 모듈을 구동해야 한다.

실행을 위한 명령은 제공된 소스의 runpath.txt의 마지막 줄에 정의되어 있다.

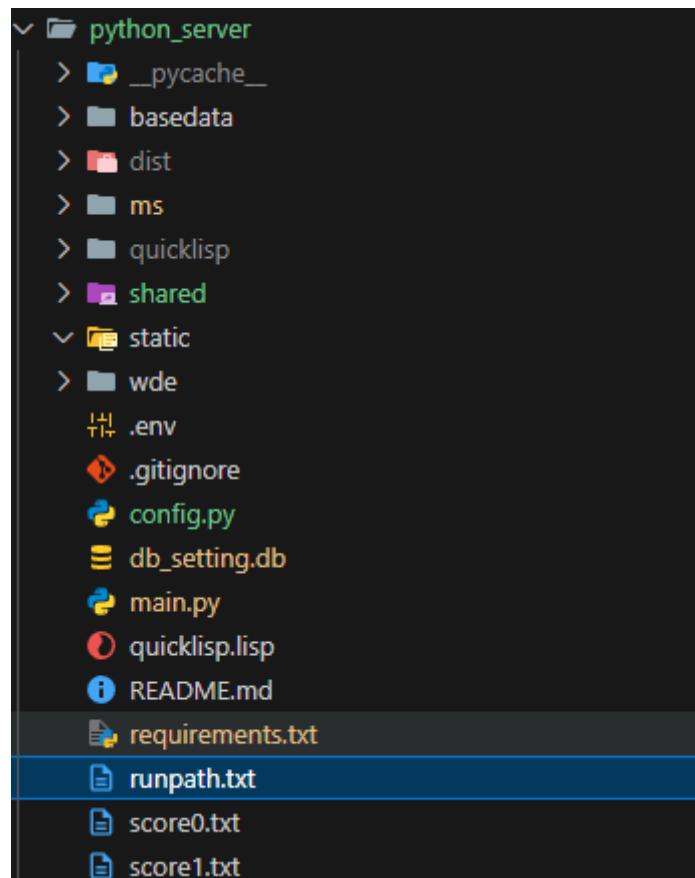
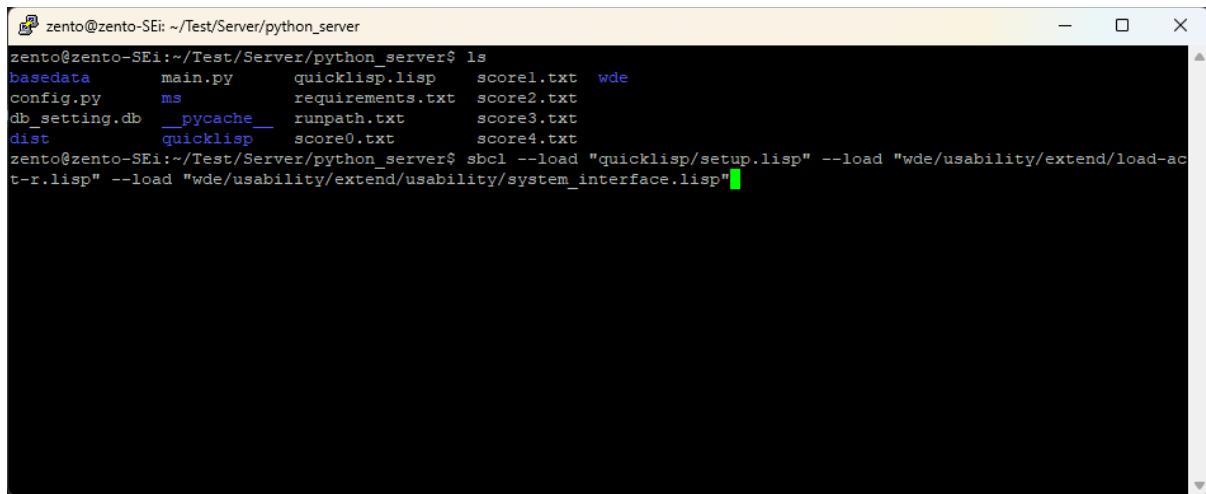


그림 6 실행 명령 제공 파일

구동 명령은 다음과 같다.

```
sbcl --load "quicklisp/setup.lisp" --load "wde/usability/extend/load-act-r.lisp" --load  
"wde/usability/extend/usability/system_interface.lisp"
```

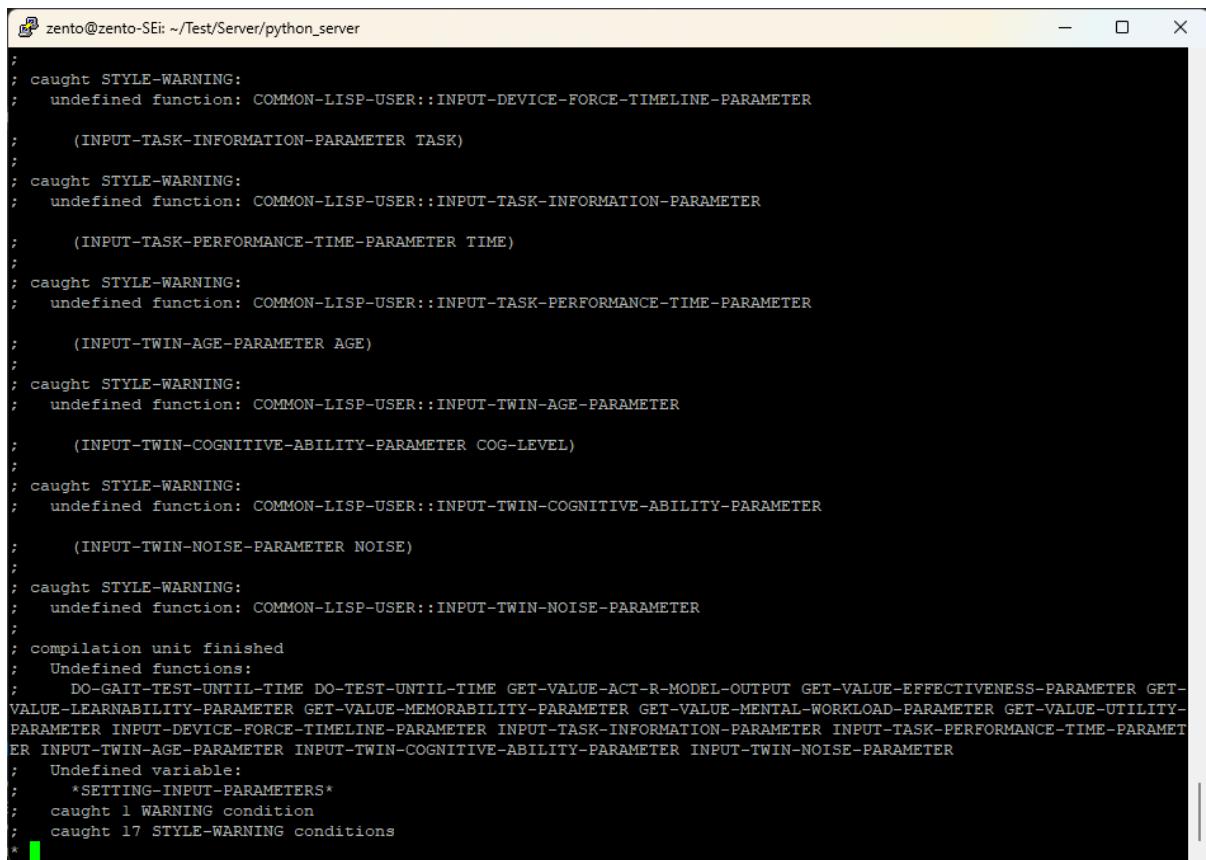
터미널에서 다음과 같이 모듈 실행 명령을 입력한다.



```
zento@zento-SEi:~/Test/Server/python_server$ ls
basedata      main.py      quicklisp.lisp    score1.txt  wde
config.py     ms          requirements.txt  score2.txt
db_setting.db __pycache__ runpath.txt      score3.txt
dist         quicklisp   score0.txt      score4.txt
zento@zento-SEi:~/Test/Server/python_server$ sbcl --load "quicklisp/setup.lisp" --load "wde/usability/extend/load-ac
t-r.lisp" --load "wde/usability/extend/usability/system_interface.lisp"
```

그림 7 ETRI 모듈 실행 명령

실행이 정상적으로 이루어지면 터미널에 다음과 같은 로그들이 출력된다.



```
; caught STYLE-WARNING:
;   undefined function: COMMON-LISP-USER::INPUT-DEVICE-FORCE-TIMELINE-PARAMETER
;
;   (INPUT-TASK-INFORMATION-PARAMETER TASK)
;
; caught STYLE-WARNING:
;   undefined function: COMMON-LISP-USER::INPUT-TASK-INFORMATION-PARAMETER
;
;   (INPUT-TASK-PERFORMANCE-TIME-PARAMETER TIME)
;
; caught STYLE-WARNING:
;   undefined function: COMMON-LISP-USER::INPUT-TASK-PERFORMANCE-TIME-PARAMETER
;
;   (INPUT-TWIN-AGE-PARAMETER AGE)
;
; caught STYLE-WARNING:
;   undefined function: COMMON-LISP-USER::INPUT-TWIN-AGE-PARAMETER
;
;   (INPUT-TWIN-COGNITIVE-ABILITY-PARAMETER COG-LEVEL)
;
; caught STYLE-WARNING:
;   undefined function: COMMON-LISP-USER::INPUT-TWIN-COGNITIVE-ABILITY-PARAMETER
;
;   (INPUT-TWIN-NOISE-PARAMETER NOISE)
;
; caught STYLE-WARNING:
;   undefined function: COMMON-LISP-USER::INPUT-TWIN-NOISE-PARAMETER
;
; compilation unit finished
; Undefined functions:
;   DO-GAIT-TEST-UNTIL-TIME DO-TEST-UNTIL-TIME GET-VALUE-ACT-R-MODEL-OUTPUT GET-VALUE-EFFECTIVENESS-PARAMETER GET-
;   VALUE-LEARNABILITY-PARAMETER GET-VALUE-MEMORABILITY-PARAMETER GET-VALUE-MENTAL-WORKLOAD-PARAMETER GET-VALUE-UTILITY-
;   PARAMETER INPUT-DEVICE-FORCE-TIMELINE-PARAMETER INPUT-TASK-INFORMATION-PARAMETER INPUT-TASK-PERFORMANCE-TIME-PARAMET
; ER INPUT-TWIN-AGE-PARAMETER INPUT-TWIN-COGNITIVE-ABILITY-PARAMETER INPUT-TWIN-NOISE-PARAMETER
; Undefined variable:
;   *SETTING-INPUT-PARAMETERS*
;   caught 1 WARNING condition
;   caught 17 STYLE-WARNING conditions
```

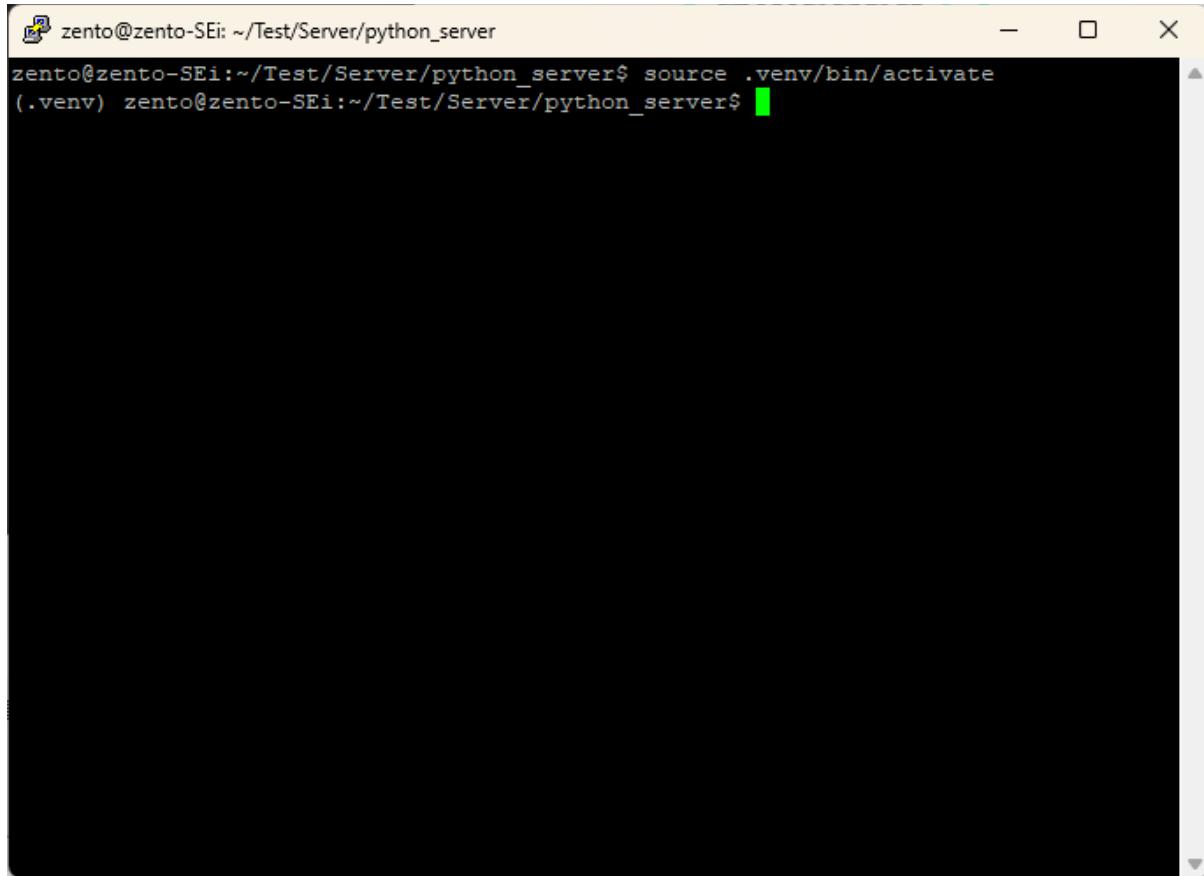
그림 8 ETRI 모듈 실행 결과

이제 frontend 서버를 구동해보겠다.

먼저 구동을 위해 가상환경을 활성화하겠다.

가상환경을 이용하지 않으면 수행하지 않아도 된다.

```
$ source .venv/bin/activate
```



The screenshot shows a terminal window with a black background and white text. At the top, it displays the user's name and session information: 'zento@zento-SEi: ~/Test/Server/python_server'. Below this, a command is entered: 'source .venv/bin/activate'. After the command is run, the terminal shows a prompt '(.venv) zento@zento-SEi:~/Test/Server/python_server\$', indicating that the virtual environment has been successfully activated. The terminal window has a standard OS X-style title bar with minimize, maximize, and close buttons.

그림 9 가상환경 활성화

서버 실행 파일은 main.py 파일이며 실행 명령은 다음과 같다.

```
$ python3 main.py
```

```
zento@zento-SEi: ~/Test/Server/python_server
interface.lisp
; in: DEFUN RUN-THE-MODEL-UNTIL-TIME
;      (DO-GAIT-TEST-UNTIL-TIME TIME)
;
; caught STYLE-WARNING:
;   undefined function: COMMON-LISP-USER::DO-GAIT-TEST-UNTIL-TIME
;
;   (DO-TEST-UNTIL-TIME TIME)
;
; caught STYLE-WARNING:
;   undefined function: COMMON-LISP-USER::DO-TEST-UNTIL-TIME
;
; compilation unit finished
; Undefined functions:
;   DO-GAIT-TEST-UNTIL-TIME DO-TEST-UNTIL-TIME
; caught 5 STYLE-WARNING conditions

!#
Server started on port 5005
 * Serving Flask app 'main'
 * Debug mode: off
Started file observer on /home/zento/Test/Server/Patient
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5005
 * Running on http://172.16.11.166:5005
Press CTRL+C to quit
```

그림 10 서버 실행 결과

위와 같은 과정을 통해 서버 실행이 완료된다.

현재 서버는 5005 번 포트를 사용해서 구동되고 있다.

3. 화면 설명

앞의 과정을 통해 서버 구동이 완료되면 웹 브라우저를 통해 해당 서버에 접속할 수 있다.

현재 매뉴얼 상에서 구동되는 Python 서버의 주소는 <http://172.16.11.166:5005/> 이다.

이 주소를 브라우저에 입력하여 접속하면 다음과 같은 화면이 출력된다.

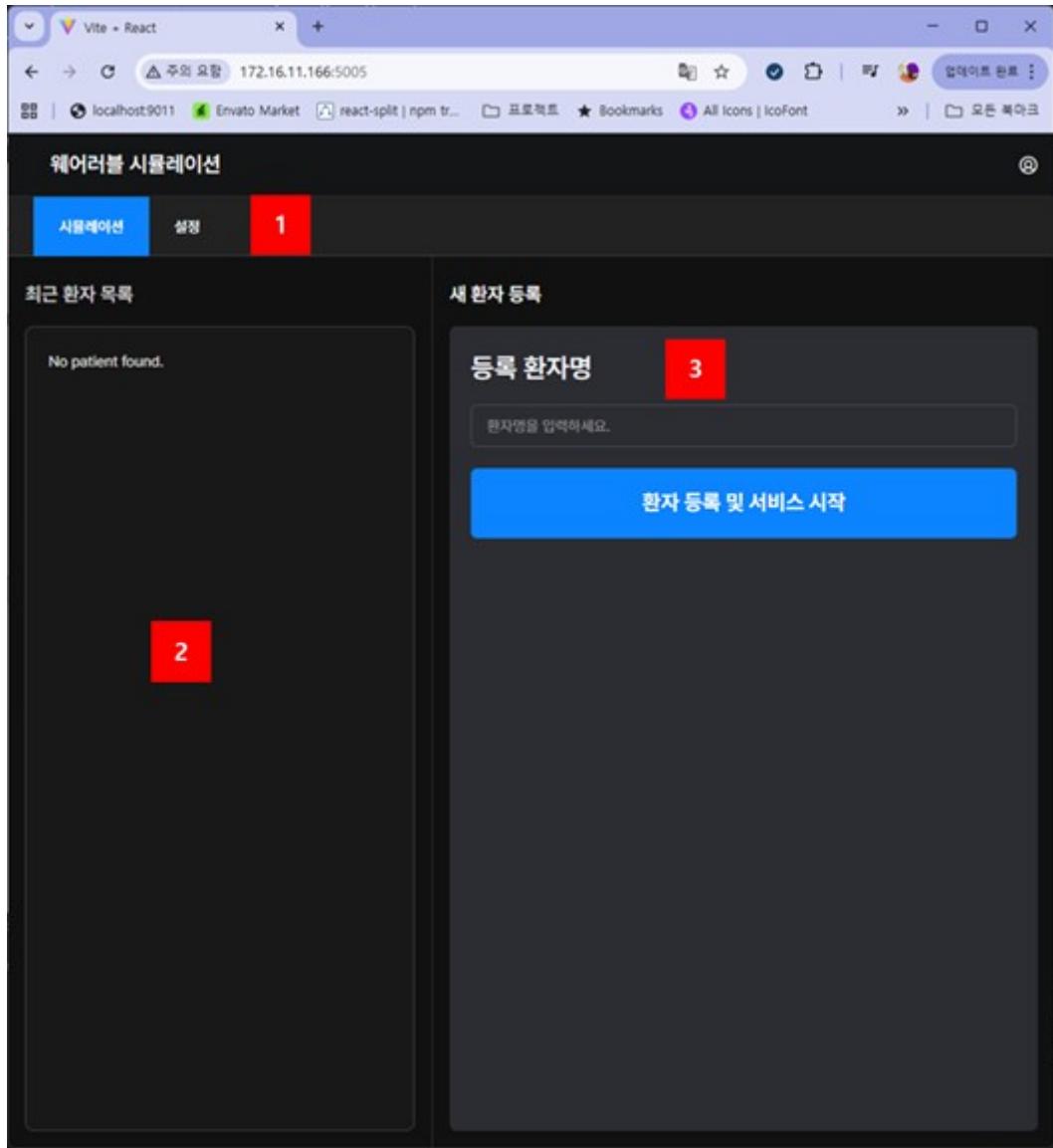


그림 11 Frontend UI 화면

처음 출력되는 화면은 시뮬레이션 서비스를 위한 환자 등록 및 선택화면이다.

1 번은 메뉴 패널이며 시뮬레이션은 시뮬레이션 서비스를 위한 메뉴이고 설정은 미리 등록할 Human 과 Device 의 설정 정보를 등록할 수 있는 페이지로 이동하기 위한 메뉴이다.

2 번은 현재까지 서비스 등록된 환자 목록이 출력되는 패널이고 3 번은 새로운 환자를 등록할 수 있는 패널이다.

서비스 수행 과정에 대해서 간단히 설명하겠다.

환자 생성

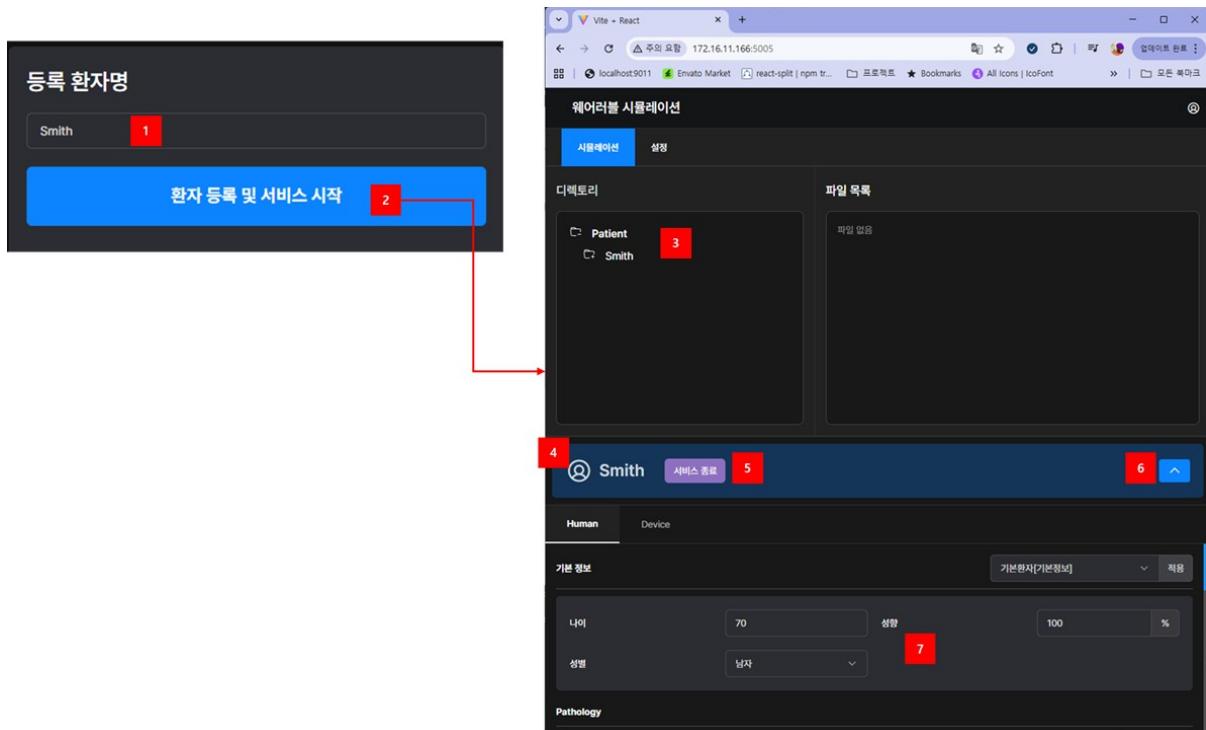


그림 12 새 환자 등록 과정

환자 등록 패널에서 1 번 등록할 환자명을 입력하고 2 번 환자 등록 및 서비스 시작 버튼을 선택한다.

그럼 서버에 해당 환자가 등록되고 Root 폴더에 환자에 대한 Index.dat 파일이 생성되며, Patient 폴더내에 등록한 환자 폴더가 생성된다.

아래는 서버에 생성된 폴더 화면이다.

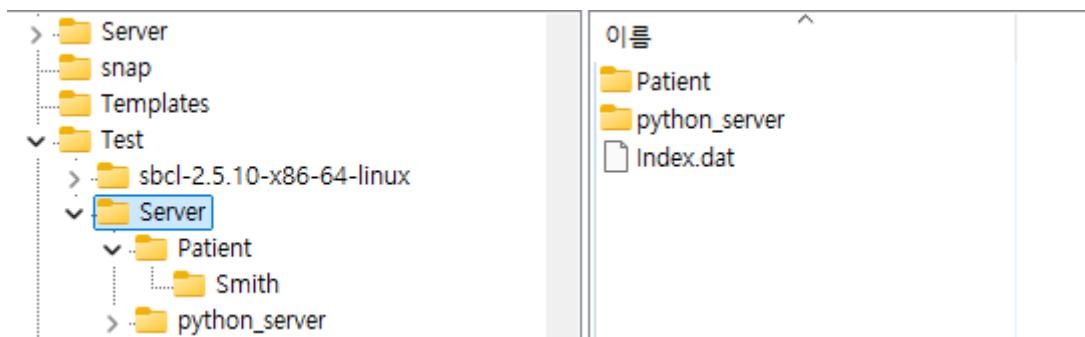


그림 13 서버 폴더 구조

Server 폴더가 Root 폴더라고 생각하면 되며, Patient 폴더가 환자관련 파일이 생성되는 폴더이다.

등록이 완료되면 시뮬레이션 화면으로 전환된다.

시뮬레이션 화면은 3 번 Patient 폴더내의 폴더 및 파일 목록이 출력되고, 4 번 선택된 환자 이름과 설정을 위한 화면이 출력된다.

이렇게 해당 환자에 대한 시뮬레이션 서비스 준비가 완료된다.

5 번 서비스 종료 버튼은 서비스를 종료하고 처음 환자 선택 화면으로 돌아가는 버튼이고 6 번은 환자 정보 설정 패널은 전체 화면으로 하는 버튼이다.

7 번은 환자에 대한 설정 정보를 입력할 수 있는 패널이다.

서비스 종료 및 선택 화면

5 번 서비스 종료 버튼을 선택하여 환자 선택 화면으로 돌아가면 다음과 같이 현재 등록된 환자 목록이 출력된다.

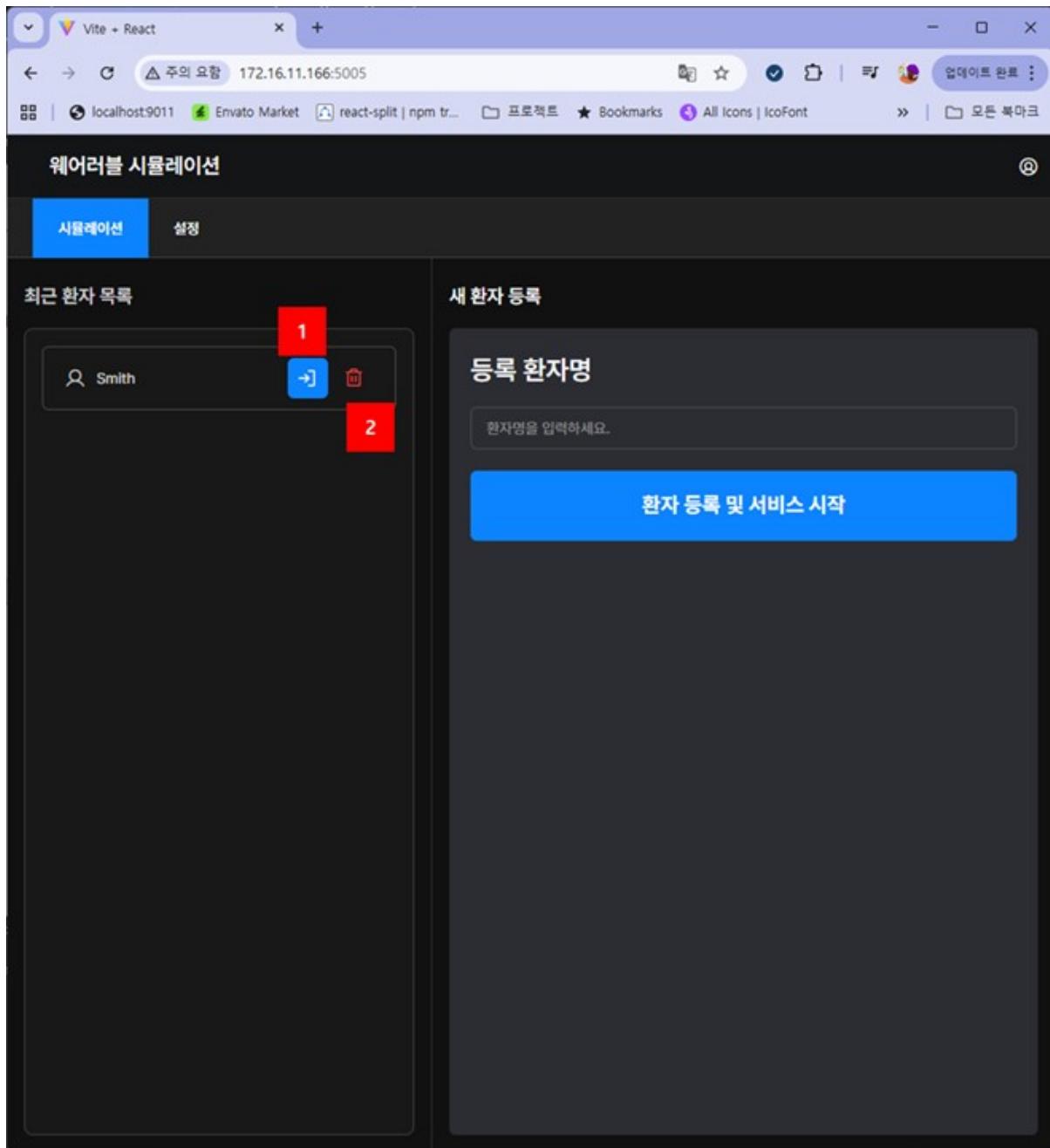


그림 14 환자 목록 출력 화면

목록에서 1 번은 해당 환자에 대한 시뮬레이션 시작을 요청하는 버튼이고 2 번은 등록된 환자를 삭제하는 버튼이다.

환자 삭제를 하면 이전에 생성된 환자 폴더 전체가 삭제된다.

1 번을 선택하여 해당 환자 시뮬레이션 서비스를 다시 시작해보겠다.

환자 정보 설정 및 XML 생성

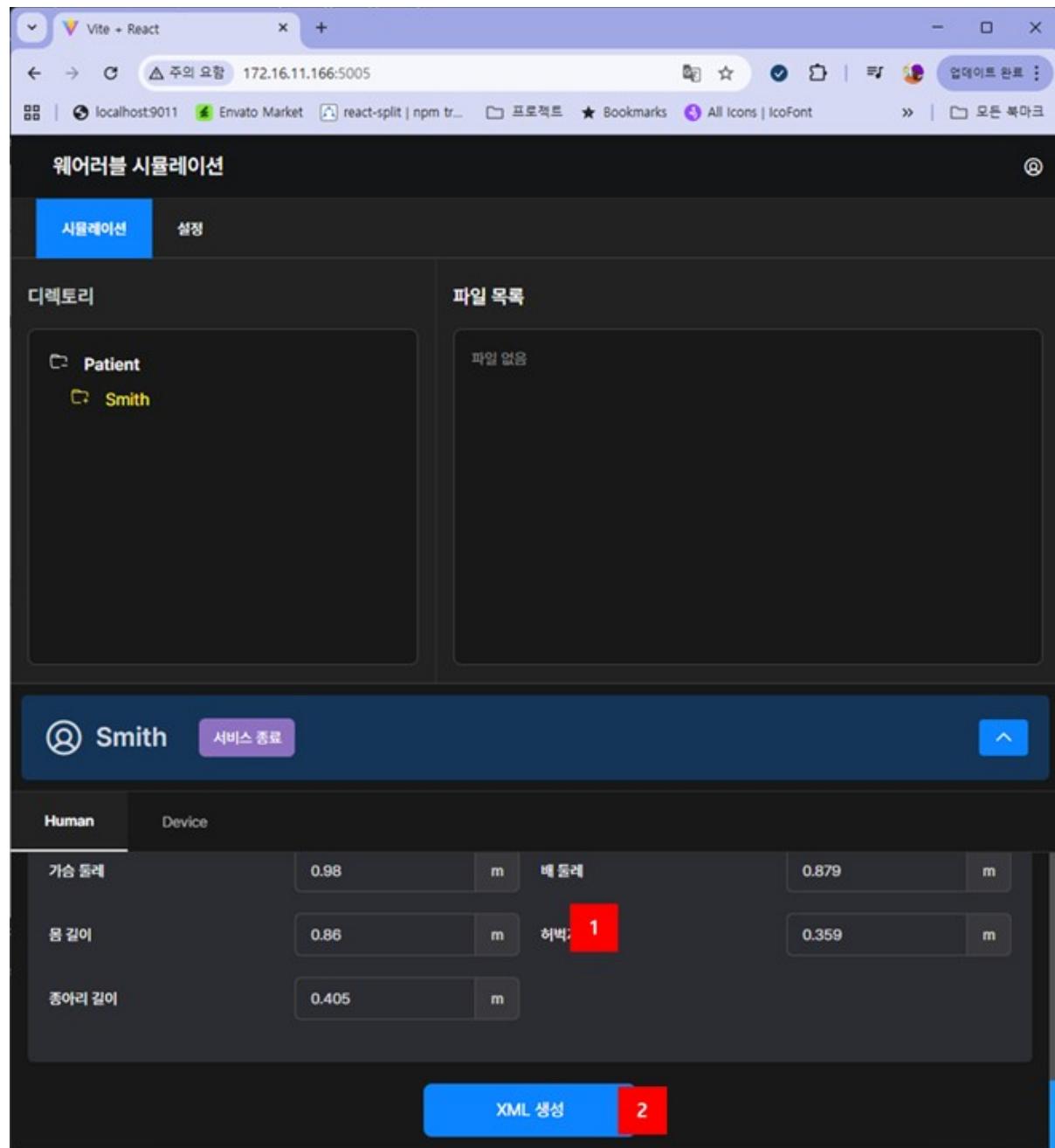


그림 15 환자 관련 XML 생성

출력된 화면에서 1 번 Human 과 Device 패널은 디폴트 정보가 기본으로 출력된다. 출력된 정보 중 변경을 원하는 정보를 입력한 후 2 번 XML 생성을 누르면 서버에 해당 환자 폴더에 환자명.xml 이 생성되고 Root 폴더에 Index.dat 파일에 설정 정보가 변경된다.

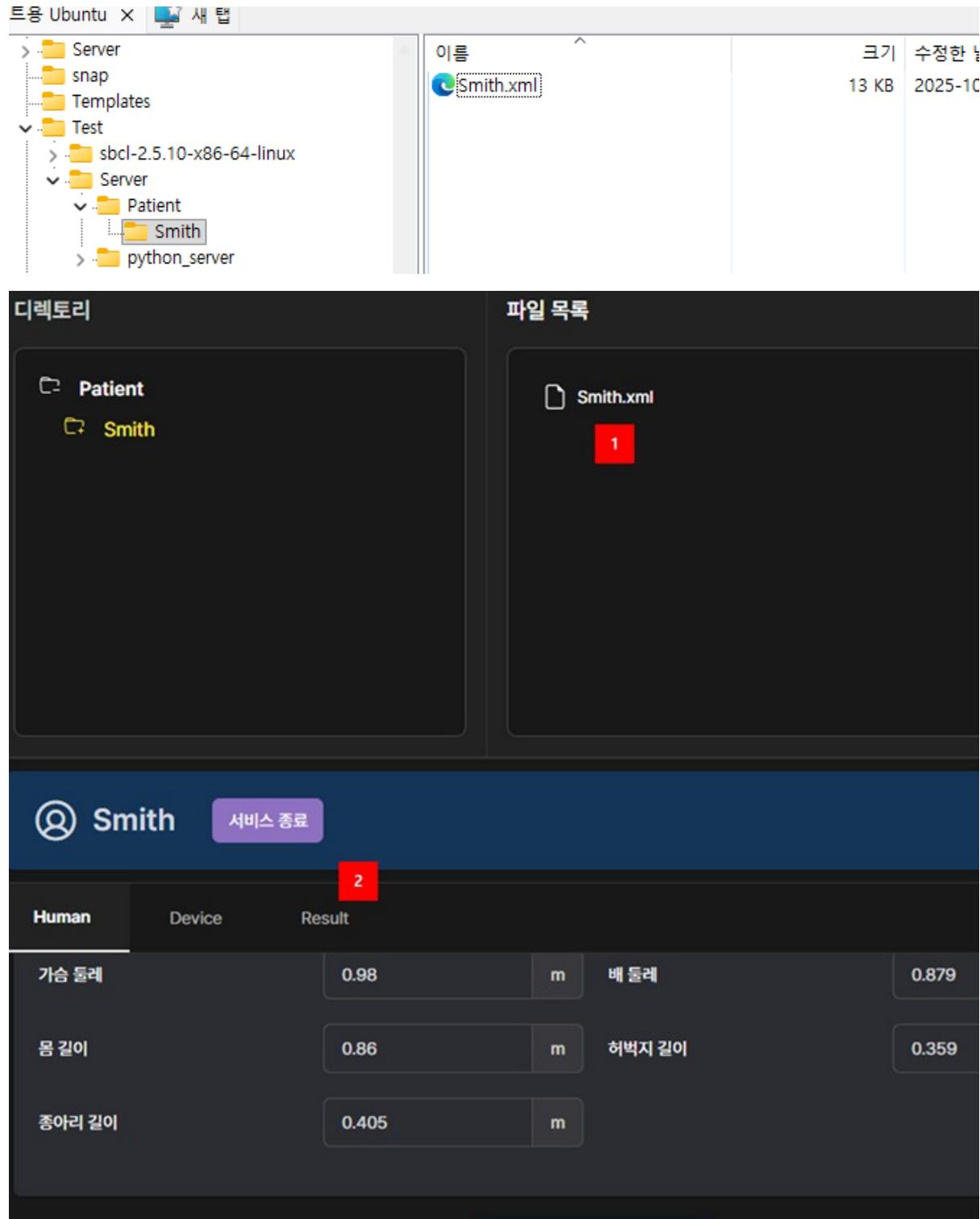


그림 16 XML 생성 결과 화면

XML 등록이 완료되면 1 번 환자 폴더에 해당 xml 이 생성된 것을 확인할 수 있으며, 아래 환자 정보 설정 패널에 Result 패널이 나타나는 것을 확인할 수 있다.

Backend 에서는 해당 환자 폴더의 환자명.xml 을 사용하여 서비스를 시작하면 된다.

결과 확인하기

위의 2 번 Result 탭을 선택하면 다음과 같은 화면이 출력된다.

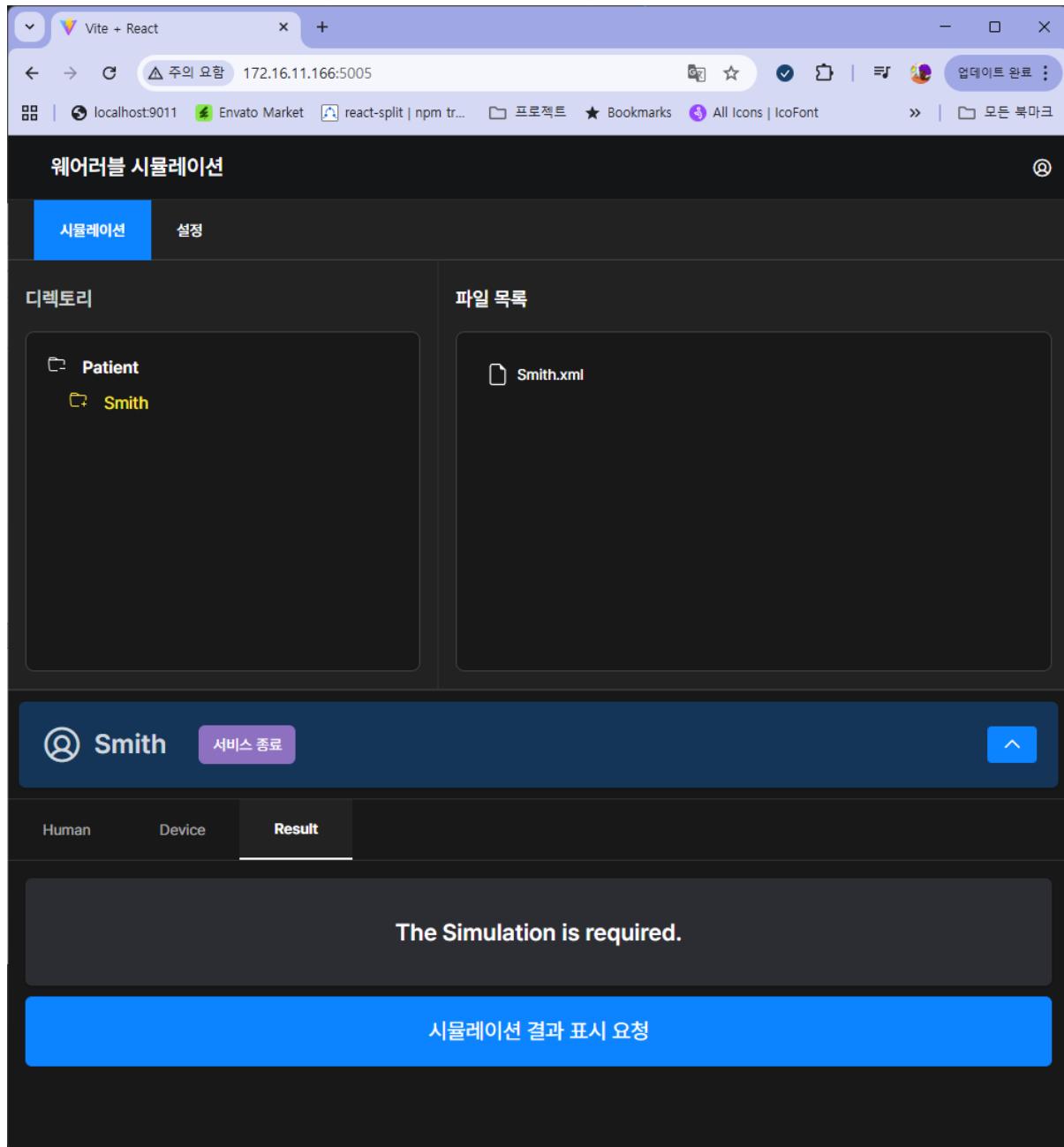


그림 17 Backend 에서 처리를 하지 않은 상태

위의 화면은 Backend 에서 제공된 xml 을 통해 서비스를 수행하기 전일 때 화면이다.

Backend 에서 서비스를 수행한 후 Result 패널의 시뮬레이션 결과 표시 요청 버튼을 선택하면 다음과 같이 해당 결과를 바탕으로 생성된 결과가 화면에 출력된다.

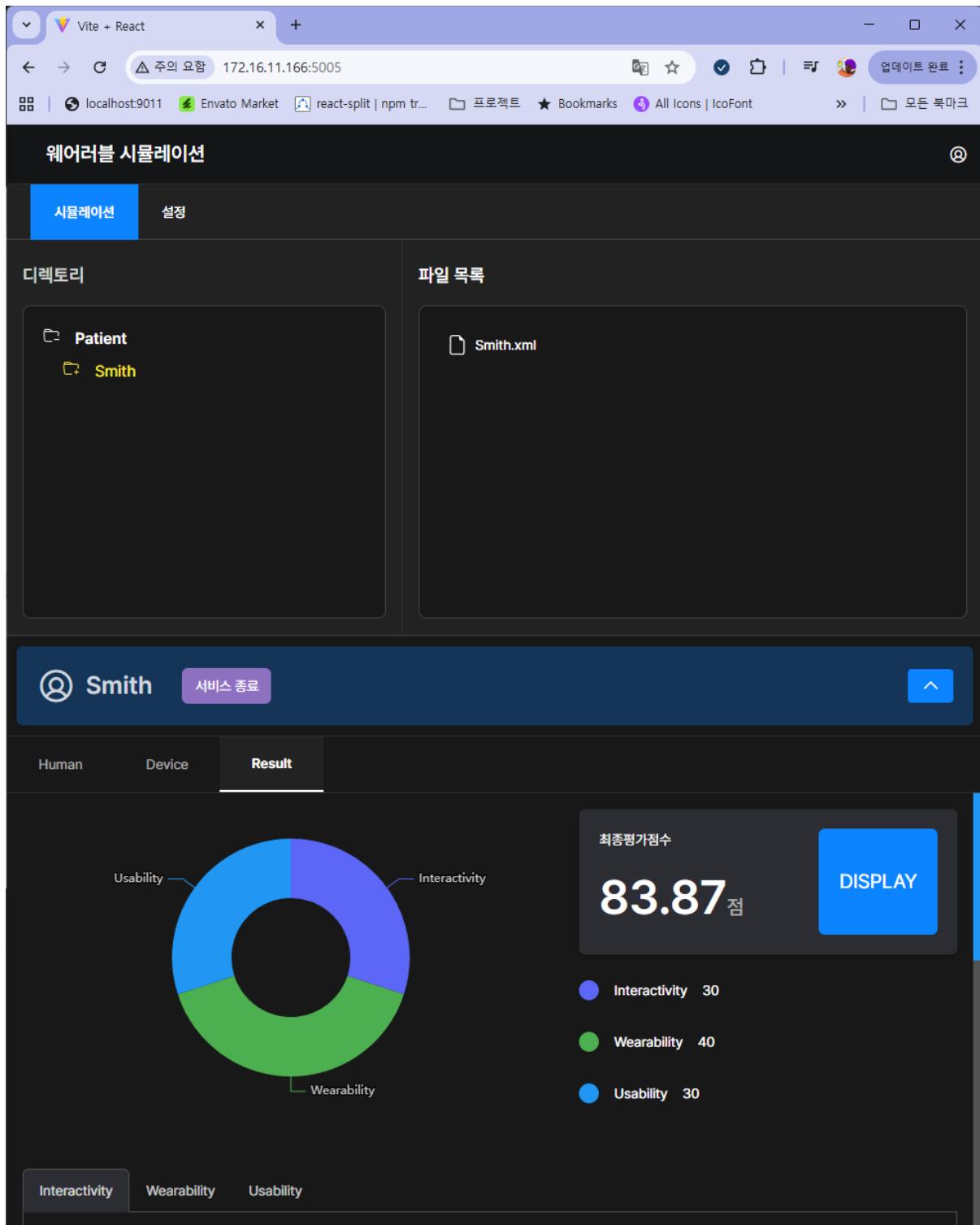


그림 18 Backend에서 처리한 후의 UI 화면

위와 같은 방식으로 frontend의 서비스를 수행할 수 있다.