SW 상세설계서

담당교수 : 김성동 교수님

제출일자 : 2024.10.16(수)

팀 : 2팀 님부스2000

1. 서론 (Introduction)

- 목적 (Purpose): 이 설계서의 목적은 AI 기반 정신과 상담 시스템의 각 구성 요소를 구체적으로 설계하여 개발자들이 소프트웨어를 효율적으로 구현할 수 있도록 돕는 것입니다.
- 범위 (Scope): 해당 설계서는 웹 애플리케이션의 프론트엔드, 백엔드, AI 서버 모듈 설계를 다룹니다.
- 참고 문서 (Reference Documents): 프로젝트 계획서, ERD, API 명세서

2. 시스템 개요 (System Overview)

- 기능 설명 (Functional Overview): 본 시스템은 AI 기반 정신과 상담 서비스로, 환자 관리, 의사 관리, 상담방 관리, 대화 관리, 프롬프트 관리 기능을 제공합니다.
- 시스템 아키텍처 (System Architecture):
  - 클라이언트: React
  - 서버: Spring Boot, FastAPI
  - 데이터베이스: MySQL
  - AI 서비스: Okestro API

3. 모듈 설계 (Module Design)

3.1. 환자 관리 모듈 (Patient Management Module)

- 기능 설명: 환자 등록, 환자 로그인, 환자 상세 조회, 환자 정보 업데이트, 환자 담당 의사 조회

- 입력: 환자 접속 코드(patient_code), 환자 정보 (patient_name, patient_gender, patient_birth)
- 출력: 환자 정보, 로그인 토큰, 환자 담당 의사 정보
- 주요 클래스 및 메서드:

```
PatientController
    patientRegistration(PatientDTO patient): 환자 등록
    patientLogin(String patientCode): 환자 로그인
  getPatientDetail(Long patientId): 환자 상세 조회
  updatePatientInfo(Long patientId, PatientDTO patient): 환자
  getDoctorInfo(Long patientId): 환자 담당 의사 조회

PatientService
  createPatient(PatientDTO patient): 환자 생성
  getPatientById(Long patientId): 환자 정보 조회
  updatePatient(Long patientId, PatientDTO patient): 환자 정보
  getDoctorInfo(Long patientId): 환자 담당 의사 조회

PatientRepository
  save(Patient patient): 환자 정보 DB에저장
  findById(Long patientId): 환자 ID로 검색
```

3.2. 의사 관리 모듈 (Doctor Management Module)

- 기능 설명: 의사 등록, 의사 로그인, 의사 정보 조회
- 입력: 의사 정보 (doctor_name, doctor_hospital, doctor_license, doctor_password)
- 출력: 등록 성공 여부, 의사 정보
- 주요 클래스 및 메서드:

```
DoctorController
  doctorRegistration(DoctorDTO doctor): 의사 등록
  doctorLogin(DoctorDTO doctor): 의사 로그인
  getDoctorInfo(Long doctorId): 의사 정보 조회

DoctorService
```

```
  createDoctor(DoctorDto doctor): 의사 생성
  getDoctorById(Long doctorId): 의사 조회

DoctorRepository
  save(Doctor doctor): 의사 정보 저장
  findById(Long doctorId): 의사 ID로 검색
```

3.3. AI 채팅 관리 모듈 (Chat Management Module)

- 기능 설명: AI 정신과 의사 펴닝에게 대화 전송 및 답변, 환자의 대화 목록 조회
- 입력: 대화 정보 (chat_id, chat_content, patient_id, chat_is_send,)
- 출력: 전송 성공 여부, 대화 내용 정보
- 주요 클래스 및 메서드:

```
ChatController
  sendChat(ChatDto chat): 메시지 전송
  getChat(Long patientId, int page, int size): 대화 내용 조회

ChatService
  sendChat(ChatDto dto): 메시지 전송
  getChatByPatientId(Long patientId, int page, int size): 대화

ChatRepository
  save(Chat chat): 대화 정보 저장
  findByPatientId(Long patientId, Pageable pageable): 환자 ID로
```

3.4. 요약보고서 관리 모듈 (Summary Management Module)

- 기능 설명: 요약보고서 생성, 조회
- 입력: X
- 출력: 성공 여부, 요약보고서 내용
- 주요 클래스 및 메서드:

```
SummaryController
  createSummary(Patient patientId): 요약보고서 생성
  getSummary(Long patientId): 요약보고서 조회
```

```
SummaryService
  createSummary(Patient patientId): 요약보고서 생성
  getSummaryByPatientId(Long patientId): 요약보고서 조회

SummaryRepository
  save(Summary summary): 프롬프트 정보 저장
  findByPatientId(Long patientId): 환자 ID로 요약보고서 검색
```

3.5. 펴닝(AI 의사) 관리 모듈 (Pyeoning Management Module)

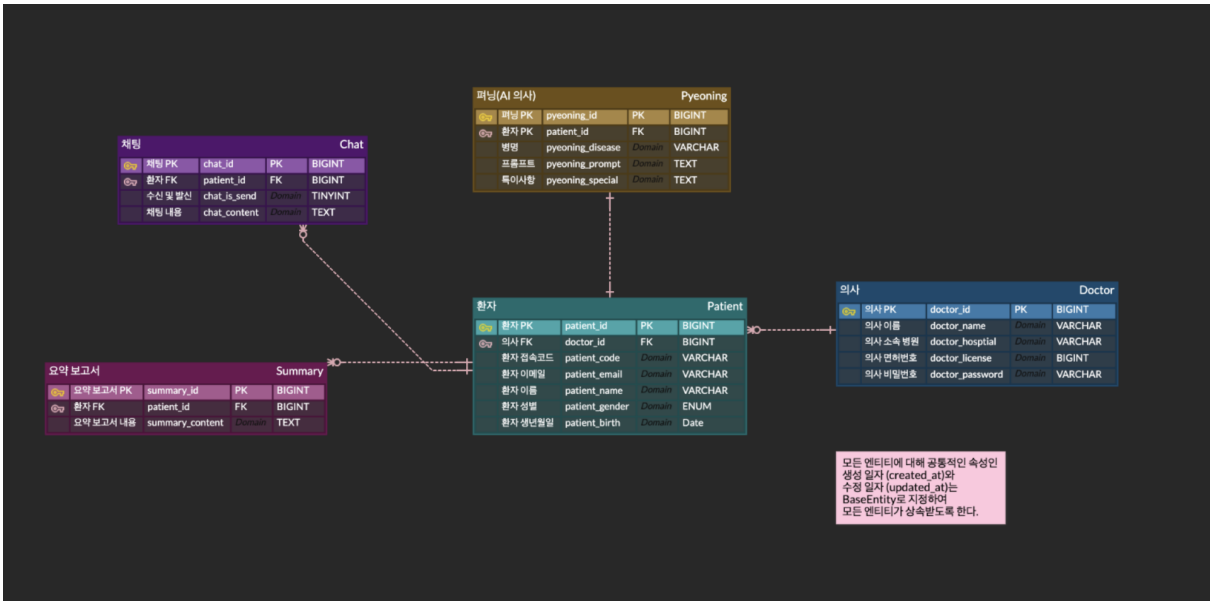
- 기능 설명: 프롬프트 생성, 조회
- 입력: 프롬프트 정보 (patient_id, pyeoning_disease, pyeoning_prompt, pyeoning_special)
- 출력: 생성 성공 여부, 프롬프트 내용
- 주요 클래스 및 메서드:

```
PyeoningController
  modifyPrompt(PyeoningDTO dto): 프롬프트 수정
  getPyeoning(Long patientId): 프롬프트 조회

PyeoningService
  modifyPrompt(PyeoningDTO dto): 프롬프트 생성
  getPromptByPatientId(Long patientId): 프롬프트 조회

PyeoningRepository
  save(Pyeoning pyeoning): 프롬프트 정보 저장
  findByPatientId(Long patientId): 환자 ID로 프롬프트 검색
```

4. 데이터베이스 설계 (Database Design)



4.1. 환자 테이블 (Patient Table)

- 테이블명: Patient
- 컬럼:
  - `patient_id` (PK, BIGINT)
  - `doctor_id` (FK)
  - `patient_code` (VARCHAR, UNIQUE)
  - `patient_email` (VARCHAR)
  - `patient_name` (VARCHAR)
  - `patient_gender` (ENUM)
  - `patient_birth` (DATE)
  - `created_at` (TIMESTAMP)
  - `updated_at` (TIMESTAMP)

4.2. 의사 테이블 (Doctor Table)

- 테이블명: Doctor
- 컬럼:
  - `doctor_id` (PK, BIGINT)
  - `doctor_name` (VARCHAR)
  - `doctor_hospital` (VARCHAR)

- `doctor_license` (BIGINT)
- `doctor_password` (VARCHAR)
- `created_at` (TIMESTAMP)
- `updated_at` (TIMESTAMP)

4.3. 채팅 테이블 (Chat Table)

- 테이블명: Chat
- 컬럼:
  - `chat_id` (PK, BIGINT)
  - `patient_id` (FK, BIGINT)
  - `chat_is_send` (TINYINT)
  - `chat_content` (TEXT)
  - `created_at` (TIMESTAMP)
  - `updated_at` (TIMESTAMP)

4.4. 요약보고서 테이블 (Summary Table)

- 테이블명: Summary
- 컬럼:
  - `summary_id` (PK, BIGINT)
  - `patient_id` (FK, BIGINT)
  - `summary_content` (TEXT)
  - `created_at` (TIMESTAMP)
  - `updated_at` (TIMESTAMP)

4.5. 펴닝(AI 의사) 테이블 (Pyeoning Table)

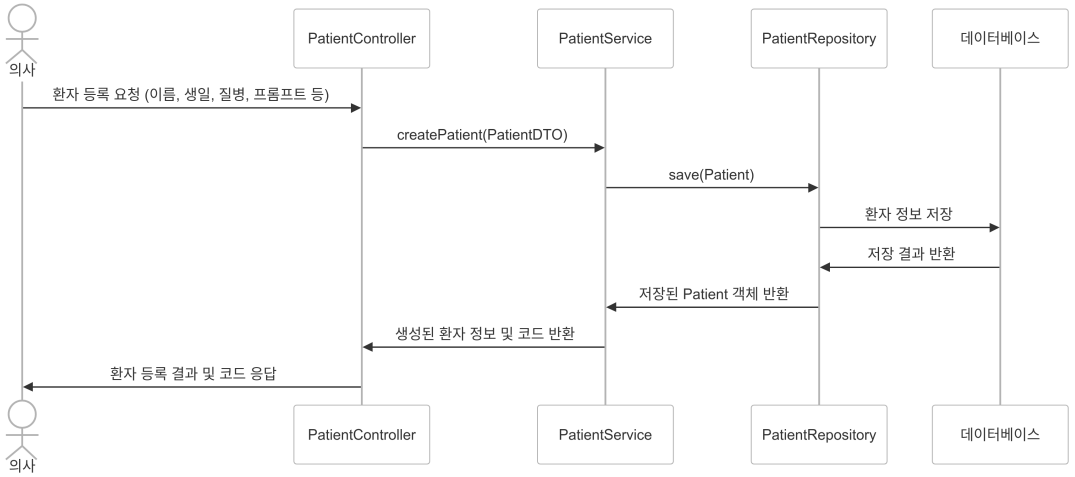
- 테이블명: Pyeoning
- 컬럼:
  - `pyeoning_id` (PK, BIGINT)
  - `patient_id` (FK, BIGINT)
  - `pyeoning_disease` (VARCHAR)

- `pyeoning_pronpt` (TEXT)
- `pyeoning_special` (TEXT)
- `created_at` (TIMESTAMP)
- `updated_at` (TIMESTAMP)

5. 메인 기능 시퀀스 다이어그램 (Sequence Diagrams)

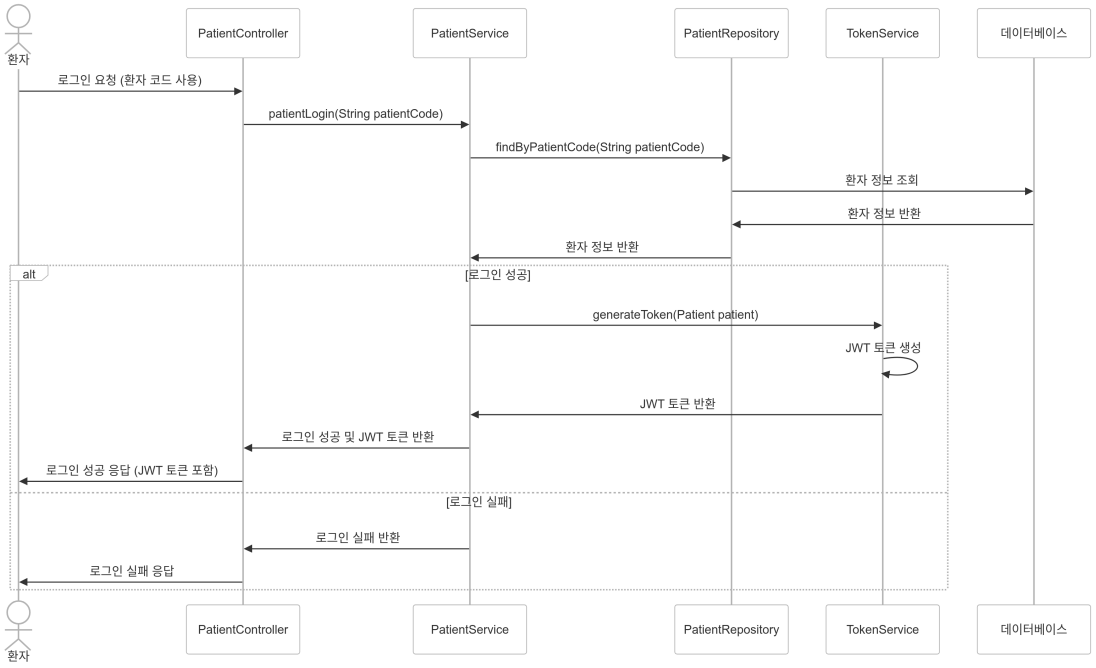
5.1. 환자 등록 시퀀스

1. 의사가 환자 등록 폼 제출
2. PatientController에서 폼 데이터 수신
3. PatientService에서 비즈니스 로직 처리 후 PatientRepository에 저장
4. 데이터베이스에 저장 완료 후 성공 메시지 반환



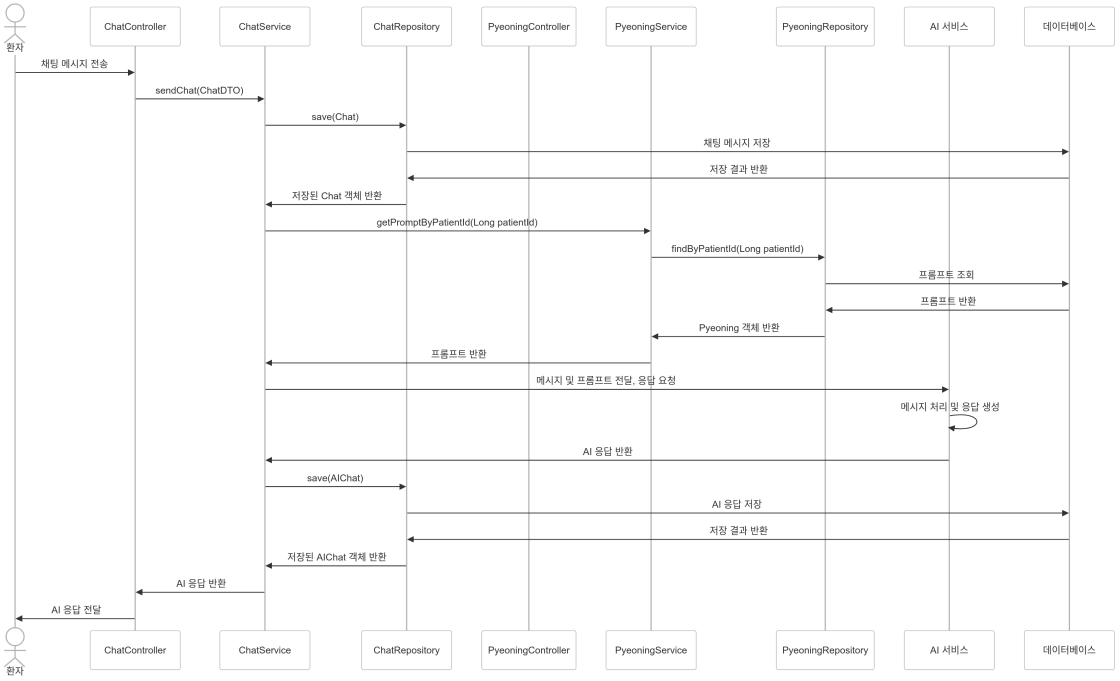
5.2. 환자 로그인 시퀀스

1. 환자가 접속코드 폼 제출
2. PatientController에서 폼 데이터 수신
3. PatientService에서 비즈니스 로직 처리 후 PatientRepository에서 접속코드로 환자 조회
4. 토큰 발급 후 응답 반환



5.3. AI 상담 채팅 시퀀스

1. 환자가 AI 챗봇에 채팅 메시지를 전송
2. ChatController에서 요청을 수신하고, 이를 적절한 Service로 매핑
3. ChatService에서 ChatRepository를 통해 DB에 접근하여 기존 채팅 내역과 프롬프트를 조회
4. 조회한 데이터를 바탕으로 AI 서버에 요청 전송
5. AI 서버에서 전달받은 데이터를 처리하여 응답 생성 후 반환
6. ChatService에서 AI 응답을 처리하고 ChatRepository를 통해 DB에 저장
7. 환자에게 AI 응답을 전송



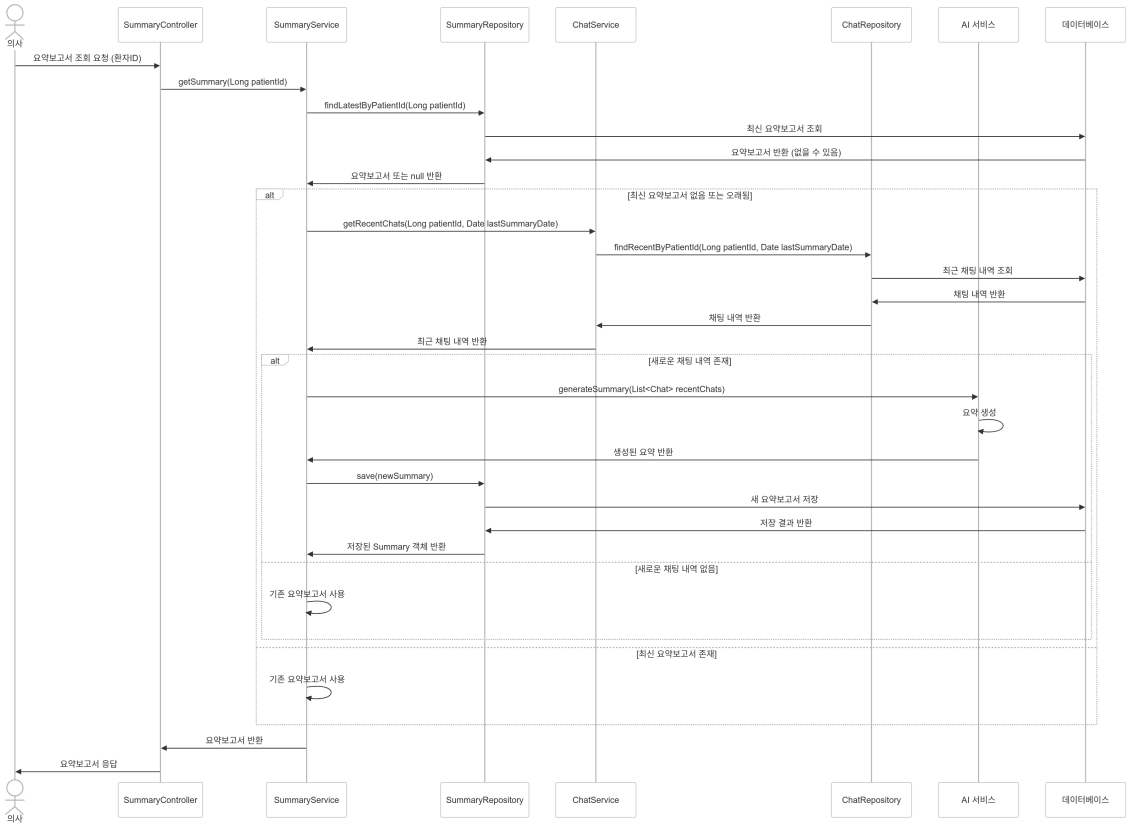
5.4. 환자 정보 조회/수정 시퀀스

1. 의사가 환자 정보 조회 또는 수정 요청을 전송
2. PatientController에서 요청을 수신하고, 적절한 Service 메소드 호출
3. PatientService에서 PatientRepository를 통해 DB에 접근하여 환자 정보 조회
4. 조회된 환자 정보를 의사에게 반환 (조회의 경우)
5. 의사가 환자 정보 수정 요청 시, PatientController를 통해 수정 데이터 전송
6. PatientService에서 수정된 데이터를 검증하고 PatientRepository를 통해 DB 업데이트

5.5. 요약 보고서 생성 시퀀스

1. 의사가 특정 환자의 요약 보고서 생성 요청
2. SummaryController에서 요청을 수신하고 SummaryService 메소드 호출
3. SummaryService에서 SummaryRepository를 통해 기존 요약 보고서 조회
4. ChatService를 통해 최근 채팅 내역 조회
5. AI 서비스에 요약 생성 요청 전송 (기존 요약 및 최근 채팅 내역 포함)

6. AI 서비스에서 새로운 요약 보고서 생성 후 반환
7. SummaryService에서 새 요약 보고서를 SummaryRepository를 통해 DB에 저장
8. 생성된 요약 보고서를 의사에게 반환



6. API 설계 (API Design)

6.1. 환자 관련 API

- **환자 등록**
  - Method : `POST`
  - URL : api/patient/signUp
  - Request :

```
{
  "patientName": "김감자",
```

```
  "patietnBrith" : "1990.12.31",
  "patientGener" : "FEMALE"
  "pyeoningDisease": "우울증",
  "pyeoningPrompt": "이 환자는~",
  "pyeoningSpecial": "고립이 어쩌구저ㄱ쩌구"
}
```

- Response :

▼ 200 (환자 등록 성공)

```
{
    "status": 200,
    "data": null,
    "message": "환자 등록에 성공했습니다."
}
```

▼ 400 (프롬프트가 비어있는 경우)

```
{
    "status": 400,
    "data": null,
    "message": "프롬프트가 비어 있습니다. 유효한 값을 입력해
}
```

- **환자 로그인**
  - Method : `POST`
  - URL : api/patient/login
  - Request :
    - RequestBody

```
{
  "patientCode": "ABCDE123"
}
```

  - Response :

▼ 200 (로그인 성공)

```
{
    "status": 200,
    "data": {
        "token" : "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJs
    },
    "message": "로그인에 성공했습니다."
}
```


▼ 404 (유효하지 않은 접속코드)


```
{
    "status": 404,
    "data": null,
    "message": "유효하지 않은 접속코드입니다."
}
```


- **환자 프롬프트 수정**
  - Method : `PUT`
  - URL : api/patient/{patientId}/modifyPrompt
  - Request :
    - RequestBody


```
{
  "patientName": "string", //수정 없는 경우 null
  "patientBirth": "date", //수정 없는 경우 null
  "pyeongDisease": "string", //수정 없는 경우 null
  "pyeongPrompt": "string", //수정 없는 경우 null
  "pyeongSpecial": "string" //수정 없는 경우 null
}
```


  - Response :

▼ 200 (환자 정보 수정 성공)


```
{
    "status": 200,
```

```
    "data": {
      "patientName": "string",
      "patientBirth": "date",
      "pyeongDisease": "string",
      "pyeongPrompt": "string",
      "pyeongSpecial": "string"
    },
    "message": "환자 정보 수정에 성공했습니다."
}
```


▼ 404 (환자 정보 찾을 수 없음)


```
{
    "status": 404,
    "data": null,
    "message": "유효하지 않은 토큰이거나, 해당 ID에 해당하는
}
```


▼ 404 (의사 정보 찾을 수 없음)


```
{
    "status": 404,
    "data": null,
    "message": "해당 ID의 환자가 DB에 존재하지 않습니다."
}
```


▼ 404 (해당 환자의 담당 의사가 아님)


```
{
    "status": 404,
    "data": null,
    "message": "해당 의사는 해당 환자의 담당 의사가 아닙니다.
```


- **환자 목록 조회**
  - Method : `GET`
  - URL : api/patient/list?page={page}&size={size}
  - Request :

- 특정 의사의 `AccessToken`
- RequestParam

```
page : 조회할 페이지 번호 (기본값 1)
size : 한 페이지에 표시할 항목 수 (기본값 10)
```

- Response :

▼ 200 (환자 목록 조회 성공 - 환자 목록이 존재하는 경우)

```
{
    "status": 200,
    "data": [{
                "patientId": "1",
          "patientName": "김환자",
          "patientGender": "FEMAIL", //enum
          "patientBirth": "2002.03,22", //String
          "patientAge": "23", //String
          "pyeongSpecial": "string" //특이사항
            },
            {
                "patientId": "2",
          "patientName": "안환자",
          "patientGender": "MAIL", //enum
          "patientBirth": "2002.07,20", //String
          "patientAge": "25", //String
          "pyeongSpecial": "string" //특이사항
            },
            ...
    ],
    "message": "환자 목록 조회에 성공했습니다."
}
```

▼ 200 (환자 목록 조회 성공 - 환자 목록이 존재하지 않는 경우)

```
{
    "status": 200,
    "data": null,
```

```
    "message": "환자 목록 조회에 성공했습니다. 환자 목록이 없
}
```

▼ 404 (해당 ID의 의사가 없는 경우)

```
{
    "status": 404,
    "data": null,
    "message": "유효하지 않은 토큰이거나, 해당 ID에 해당하는
}
```

▼ 500 (내부 서버 오류)

```
{
    "status": 500,
    "data": null,
    "message": "내부 서버 오류가 발생했습니다."
}
```

- **환자 상세 조회**
  - Method : `GET`
  - URL : api/patient/{patientId}/detail
  - Request :
    - 특정 환자의 `AccessToken`
  - Response :

▼ 200 (환자 상세 조회 성공)

```
{
    "status": 200,
    "data": {
          "patientName": "김감자",
          "patientGender": "여",
          "patientBirth": "1991.04.11",
          "pyeoningDisease": "우울증",
          "pyeoningPrompt": "이 환자는... 더보기",
          "pyeoningSpecial": "우울증과 불안... 더보기"
```

```
},
    "message": "환자 상세 조회에 성공했습니다."
}
```

▼ 404 (해당 ID의 환자가 없는 경우)

```
{
    "status": 404,
    "data": null,
    "message": "해당 ID의 환자가 DB에 존재하지 않습니다."
}
```

▼ 404 (해당 ID의 의사가 없는 경우)

```
{
    "status": 404,
    "data": null,
    "message": "해당 ID의 의사가 DB에 존재하지 않습니다."
}
```

▼ 404 (해당 환자의 담당 의사가 아님)

```
{
    "status": 404,
    "data": null,
    "message": "해당 의사는 해당 환자의 담당 의사가 아닙니다.
```

▼ 500 (내부 서버 오류)

```
{
    "status": 500,
    "data": null,
    "message": "내부 서버 오류가 발생했습니다."
}
```

- **담당 의사 정보 조회**
  - Method : `GET`

- URL : api/patient/doctorInfo
- Request :
  - 특정 환자의 `AccessToken`
- Response :

▼ 200 (의사 정보 조회 성공)

```
{
    "status": 200,
    "data": {
     "doctorName": "김정은",
     "doctorHospital": "참좋은병원"
    },
    "message": "의사 정보 조회에 성공했습니다."
}
```

▼ 404 (존재하지 않는 환자)

```
{
    "status": 404,
    "data": null,
    "message": "유효하지 않은 토큰이거나, 해당 ID에 해당하는
}
```

▼ 404 (담당 의사가 존재하지 않는 경우)

```
{
    "status": 404,
    "data": null,
    "message": "해당 환자의 담당의사가 존재하지 않습니다."
}
```

6.2. 의사 관련 API

- **의사 등록**
  - Method : `POST`
  - URL : api/doctor/register

- Request :
  - RequestBody

```
{
  "doctorName": "string",
  "doctorHospital": "string",
  "doctorLicense": "string",
  "doctorPassword": "string"
}
```

- Response :

▼ 200 (의사 등록 성공)

```
{
    "status": 200,
    "data": null,
    "message": "의사 등록에 성공했습니다."
}
```

▼ 400 (Bad Request)

```
{
    "status": 400,
    "data": null,
    "message": "의사 등록에 실패했습니다."
}
```

- **의사 면허 번호 중복검사**
  - Method : `GET`
  - URL : api/doctor/license/check?licenseNumber={licenseNumber}
  - Request :
    - RequestParam

```
licenseNumber : 의사 면허 번호
```

  - Response :

▼ 200 (사용 가능한 면허 번호)

```
{
  "status": 200,
  "data": null,
  "message": "사용 가능한 면허 번호입니다."
}
```

▼ 400 (이미 등록된 면허 번호)

```
{
  "status": 400,
  "data": null,
  "message": "이미 등록된 면허 번호입니다."
}
```

- **의사 로그인**
  - Method : `GET`
  - URL : api/doctor/login
  - Request :
    - RequestBody

```
{
  "doctorLicense": "string",
  "doctorPassword": "string"
}
```

  - Response :

▼ 200 (의사 로그인 성공)

```
{
    "status": 200,
    "data": {
        "accessToken": "eyJhbGciOiJIUzI1NiJ9.eyJzdWI
    },
```

```
    "message": "의사 로그인에 성공했습니다."
}
```

▼ 401 (비밀번호 일치하지 않음)

```
{
    "status": 401,
    "data": null,
    "message": "비밀번호가 일치하지 않습니다.."
}
```

▼ 404 (면호 번호에 해당하는 의사 없음)

```
{
    "status": 404,
    "data": null,
    "message": "면호 번호에 해당하는 의사가 없습니다.."
}
```

6.3. 채팅 관련 API

- **환자가 채팅메시지 전송, 답변 수신**
  - Method : `POST`
  - URL : api/chat/send
  - Request :
    - 특정 환자의 `AccessToken`
    - RequestBody

```
{
  "chatContent": "string"
}
```

  - Response :

▼ 200 (메세지 전송 및 AI 응답 생성에 성공)

```
{
    "status": 200,
    "data": {
            "chatId": 1,
          "chatContent": "string",
            "createdAt": "timestamp"
    },
    "message": "메세지 전송 및 AI 응답 생성에 성공했습니다.'
}
```


▼ 404 (환자 정보 찾을 수 없음)


```
{
    "status": 404,
    "data": null,
    "message": "유효하지 않은 토큰이거나, 해당 ID에 해당하는
}
```


▼ 502 (AI 서버와의 통신 실패)


```
{
    "status": 404,
    "data": null,
    "message": "AI 서버와의 통신에 실패했습니다. 잠시 후 다시
}
```


▼ 504 (AI 서버 시간초과)


```
{
  "status": 504,
  "data": null,
  "message": "AI 서버 응답 시간이 초과되었습니다. 잠시 후 다
}
```


- **대화 내용 전체 조회**
  - Method : `GET`
  - URL : api/chat/history?page={page}&size={size}

- Request :
  - `AccessToken`
  - RequestParam

```
page : 조회할 페이지 번호 (기본값 1)
size : 한 페이지에 표시할 항목 수 (기본값 10)
(의사가 환자 대화 조회시 필요!) patientId : 환자 id
```

- Response :

▼ 200 (대화 내용 조회 성공 - 대화 내용이 존재하는 경우)

```
{
    "status": 200,
    "data": [
    {
      "chatId": 1,
      "chatIsSend": 1, //수신이면 0, 발신이면 1
      "chatContent": "string",
      "createdAt": "datetime"
    },
        {
      "chatId": 1,
      "chatIsSend": 0, //수신이면 0, 발신이면 1
      "chatContent": "string",
      "createdAt": "datetime"
    },
    ...
  ],
    "message": "대화 내용 조회에 성공했습니다."
}
```

▼ 200 (대화 내용 조회 성공 - 대화 내용이 존재하지 않는 경우)

```
{
    "status": 200,
    "data": null,
```

```
    "message": "대화 내용 조회에 성공했습니다. 대화 내용이 존
}
```


▼ 403(의사가 환자의 담당 의사가 아닌 경우)


```
{
  "status": 403,
  "data": null,
  "message": "해당 환자의 담당 의사가 아닙니다."
}
```


▼ 404 (환자 정보 찾을 수 없음)


```
{
    "status": 404,
    "data": null,
    "message": "유효하지 않은 토큰이거나, 해당 ID에 해당하는
}
```


▼ 404 (의사 정보 찾을 수 없음)


```
{
    "status": 404,
    "data": null,
    "message": "유효하지 않은 토큰이거나, 해당 ID에 해당하는
}
```


▼ 500 (내부 서버 오류)


```
{
    "status": 500,
    "data": null,
    "message": "내부 서버 오류가 발생했습니다."
}
```


6.4. 요약 보고서 관련 API

- **요약보고서 생성**

- Method : `POST`
- URL : api/patient/signUp
- Request :
  - 특정 환자의 `AccessToken`
- Response :

▼ 200 (요약보고서 생성 성공)

```
{
    "status": 200,
    "data": {
            "summaryId": 1,
          "summaryContent": "해당 환자는... 더보기",
            "createdAt": "timestamp"
    },
    "message": "요약보고서 생성에 성공했습니다."
}
```

▼ 404 (환자 정보 찾을 수 없음)

```
{
    "status": 404,
    "data": null,
    "message": "유효하지 않은 토큰이거나, 해당 ID에 해당하는
}
```

- **요약보고서 조회**
  - Method : `GET`
  - URL : api/summary/patientSummary/{patientId}
  - Request :
    - 특정 환자의 `AccessToken`
  - Response :

▼ 200 (요약보고서 조회 성공)

```
{
    "status": 200,
    "data": {
            "summaryId": 1,
          "summaryContent": "해당 환자는... 더보기",
            "createdAt": "timestamp"
    },
    "message": "요약보고서 조회에 성공했습니다."
}
```

▼ 400 (요약보고서 조회 실패)

```
{
    "status": 400,
    "data": null,
    "message": "요약보고서 조회에 실패했습니다."
}
```

▼ 404 (환자 정보 찾을 수 없음)

```
{
    "status": 404,
    "data": null,
    "message": "유효하지 않은 토큰이거나, 해당 ID에 해당하는
}
```

6.5. AI 관련 API

- **의사 AI 와의 채팅 기능**
  - Method : `POST`
  - URL : api/doctor-ai/chat
  - Request :
    - RequestBody

```
{
  "newChat": "string", // 질문하는 메세지
```

```
    "chatHistory": [
    {"sender": "김혜진", "message": "나 지금 너무 힘들어"
    {"sender": "펴닝", "message": "혹시 무슨 이유로 힘드실
    ...
    ]
  "prompt": "string"
}
```

- Response :

▼ 200 (AI 응답 생성)

```
{
  "status": 200,
  "data": {
    "response": "펴닝의 답장", //String
  },
  "message": "AI 응답이 성공적으로 생성되었습니다."
}
```

▼ 400 (새로운 질문 없음)

```
{
    "status" : 400,
    "data" : null,
    "message" : "새로운 질문이 없습니다."
}
```

▼ 500 (내부 서버 오류)

```
{
  "status": 500,
  "data": null,
  "message": "내부 서버 오류가 발생했습니다. AI 모델 처리 중
}
```

- **요약 보고서 생성**
  - Method : `POST`

- URL : api/doctor-ai/summarize
- Request :
  - RequestBody

```
{
  "disease": "우울증", // 병명
    "chat_history": [
    {"sender": "김혜진", "message": "나 지금 너무 힘들어"
    {"sender": "펴닝", "message": "혹시 무슨 이유로 힘드실
    ...
    ]
}
```

- Response :

▼ 200 (요약 보고서 생성 성공)

```
{
    "status": 200,
    "data": {
    "summary": "환자는 현재 어떤 상태이고~ 어떤 대화를 했고~
    },
    "message": "요약 보고서가 성공적으로 생성되었습니다."
}
```

▼ 400 (대화 내용 필요)

```
{
    "status": 400,
    "data": null,
    "message": "요약을 위한 대화 내용이 필요합니다."
}
```

7. 에러 처리 및 로깅 (Error Handling and Logging)

에러 처리 방안:

- 전역 예외 처리기 구현: @ExceptionHandler를 사용하여 모든 컨트롤러에서 발생하는 예외를 일관되게 처리
- 사용자 정의 예외 클래스 생성: 비즈니스 로직에 특화된 예외 정의 (예: PatientNotFoundException, DoctorAssignmentException 등)
- 에러 응답 형식 표준화: HTTP 상태 코드, 에러 메시지, 에러 코드 등을 포함한 일관된 형식의 에러 응답 제공

로깅 시스템:

- SLF4J와 Logback을 사용하여 애플리케이션 로그 기록
- 로그 포맷 정의: 타임스탬프, 로그 레벨, 클래스명, 메시지 등을 포함한 로그 포맷 설정
- 로그 저장 및 관리: 일별 로그 파일 생성 및 로그 로테이션 설정

8. 보안 설계 (Security Design)

- 인증 및 권한 관리: JWT 토큰을 사용한 사용자 인증 및 권한 부여
- 암호화: 비밀번호 해시 방식 (bcrypt 사용)

9. 테스트 계획 (Test Plan)

- 유닛 테스트: 각 서비스 클래스에 대한 단위 테스트 구현
- 통합 테스트: API 엔드포인트에 대한 통합 테스트 구현
- 성능 테스트: 대량의 동시 사용자 접속 시 시스템 성능 테스트
- AI 모델 테스트: 다양한 입력에 대한 AI 모델의 응답 정확도 테스트

10. 배포 계획 (Deployment Plan)

- 배포 환경: AWS EC2 인스턴스 사용