

설스터디 프론트엔드 API 연동 가이드

기본 정보

항목	값
개발 서버	http://localhost:8080
운영 서버	https://api.seolstudy.cloud
Swagger 문서	{서버주소}/swagger-ui/index.html
인증 방식	JWT Bearer Token
기본 Content-Type	application/json (파일 업로드는 multipart/form-data)

1. 인증 (로그인)

로그인

```
POST /auth/login
Content-Type: application/json
{ "username": "mentee1", "password": "1234" }
```

주의: 토큰은 응답 Body가 아니라 응답 Header에 있습니다!

```
응답 Status: 200 OK
응답 Header → Authorization: Bearer eyJhbGciOiJIUzI1NiJ9...
응답 Body: 없음 (빈 응답)
```

axios 예시

```
const res = await axios.post('/auth/login', {
  username: 'mentee1',
  password: '1234',
});

// 헤더에서 토큰 꺼내기
const token = res.headers['authorization'] as string; // "Bearer eyJ...
localStorage.setItem('token', token);
```

fetch 예시

```
const res = await fetch('/auth/login', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ username: 'mentee1', password: '1234' }),
});

const token = res.headers.get('Authorization'); // "Bearer eyJ..."  
if (token) localStorage.setItem('token', token);
```

이후 모든 API 요청

```
// axios 공통 설정
axios.defaults.headers.common['Authorization'] = localStorage.getItem('token') ?? '';

// 또는 요청마다
axios.get('/users/me', {
  headers: { Authorization: localStorage.getItem('token') ?? '' },
});
```

로그아웃

서버 API 없음. `localStorage.removeItem('token')` 하고 로그인 페이지로 이동하면 됩니다.

테스트 계정

아이디	비밀번호	역할	매핑
mentor1	1234	멘토	mentee1, mentee2의 멘토
mentee1	1234	멘티	mentor1의 멘티
mentee2	1234	멘티	매핑 없음

2. 사용자 정보

내 정보 조회

```
GET /users/me

{
  "id": 2,
  "username": "mentee1",
  "name": "멘티1",
  "nickname": "멘티닉네임",
  "role": "MENTEE",
```

```
        "profileImgUrl": "https://...",
        "isAIEnabled": true
    }
```

내 정보 수정

```
PATCH /users/me
Content-Type: application/json

{ "name": "새이름", "nickname": "새닉네임" }
```

프로필 이미지 업로드

```
PATCH /users/me/profile
Content-Type: multipart/form-data
```

- file: 이미지 파일 (jpg/png)

FCM 토큰 등록 (푸시 알림용)

```
POST /users/me/fcm-token
Content-Type: application/json

{ "fcmToken": "디바이스FCM토큰값" }
```

3. 멘토-멘티 관계

[멘토] 내 멘티 목록

```
GET /mentor/mentees?page=0&size=20
```

[멘티] 내 멘토 조회

```
GET /mentee/mentor
```

4. 플래너

[멘티] 플래너 생성

```
POST /plans
{ "planDate": "2026-02-10", "dailyMemo": "오늘 화이팅!" }
```

날짜별 플래너 조회

```
GET /plans?year=2026&month=2&day=10
```

멘토가 멘티 플래너 볼 때:

```
GET /plans?year=2026&month=2&day=10&menteeId=2
```

응답:

```
{
  "id": 1,
  "planDate": "2026-02-10",
  "totalStudyTime": 9000,
  "totalStudyTimeFormatted": "02: 30: 00",
  "dailyMemo": "오늘은 수학에 집중하자!",
  "mentorFeedback": null,
  "menteeId": 2,
  "tasks": [
    {
      "id": 1,
      "title": "수학 이차방정식 문제풀이",
      "subject": "MATH",
      "isMentorChecked": false,
      "actualStudyTimeFormatted": "01: 30: 00",
      "isMandatory": true,
      "taskDate": "2026-02-10"
    }
  ],
  "createdAt": "2026-02-10T00:00:00"
}
```

totalStudyTime은 초 단위 정수, totalStudyTimeFormatted는 "HH: MM: SS" 문자열입니다.

월간 캘린더 조회

```
GET /plans/calendar?year=2026&month=2&page=0&size=31
```

멘토가 멘티 캘린더 볼 때:

```
GET /plans/calendar?menteeId=2&year=2026&month=2
```

필터 옵션:

- subject=MATH — 특정 과목만
- incompleteOnly=true — 미완료 과제만

주간 캘린더 조회

```
GET /plans/calendar/weekly?date=2026-02-10&page=0&size=7
```

멘토가 멘티 것 볼 때: &menteeId=2 추가

[멘토] 플래너 수정

```
PUT /plans/{planId}  
{ "daiLyMemo": "수정된 메모" }
```

[멘토] 플래너 삭제

```
DELETE /plans/{planId}
```

5. 과제 (Task)

[멘토] 과제 출제

```
POST /tasks/{menteeId}  
{  
  "subject": "MATH",  
  "title": "수학 이차방정식 문제풀이",  
  "weekNumber": 1,  
  "startDate": "2026-02-10",  
  "endDate": "2026-02-28",  
  "daysOfWeek": ["MON", "WED", "FRI"],  
  "weaknessId": 1,  
  "dayContents": [  
    { "day": "MON", "contentId": 5 },  
    { "day": "WED", "contentId": 6 },  
    { "day": "FRI", "contentId": 7 }  
  ]  
}
```

- `subject`: KOREAN / ENGLISH / MATH / OTHER
- `daysOfWeek`: MON ~ SUN
- `weekNumber`: 프론트에서 직접 계산해서 보내야 합니다. 서버는 이 값을 그대로 저장만 합니다 (표시용). `startDate/endDate`로 자동 계산되지 않으니 UI에서 "몇 주차"인지 선택하거나 계산해서 넘겨주세요.
- `weaknessId`, `dayContents`: 선택사항

[멘토] 할 일 추가

```
POST /tasks  
{  
  "date": "2026-02-10",  
  "title": "영어 단어 복습",
```

```
        "subject": "ENGLISH"  
    }
```

과제 상세 조회

GET /tasks/{taskId}

응답:

```
{  
    "id": 1,  
    "subject": "MATH",  
    "title": "수학 이차방정식 문제풀이",  
    "status": "TODO",  
    "actualStudyTime": 5400,  
    "actualStudyTimeFormatted": "01: 30: 00",  
    "taskDate": "2026-02-10",  
    "isMandatory": true,  
    "isMentorChecked": false,  
    "recurringGroupId": "uuid...",  
    "weekNumber": 1,  
    "timerStatus": "STOPPED",  
    "weaknessId": 1,  
    "contentId": 5,  
    "fileName": "이차방정식 기본 개념 정리",  
    "fileUrl": "https://s3.../uuid.pdf",  
    "dailyPlanId": 1,  
    "menteeId": 2,  
    "submitted": false,  
    "createdAt": "2026-02-05T10: 00: 00",  
    "updatedAt": "2026-02-05T10: 00: 00"  
}
```

핵심 필드 설명:

- `isMandatory: true` → 멘토가 낸 과제 (핀 아이콘 표시), `false` → 멘티가 추가한 할 일
- `status: TODO` → `IN_PROGRESS` → `DONE`
- `submitted: true` → 과제 제출 완료 (`status`가 `DONE`이면 `true`)
- `timerStatus: STOPPED / RUNNING`
- `actual StudyTime`: 누적 공부시간(초), `actual StudyTimeFormatted: "HH: MM: SS"` 포맷

과제 수정

PUT /tasks/{taskId}

```
{ "title": "수정된 제목", "subject": "MATH", "status": "IN_PROGRESS" }
```

- 필수 과제(`isMandatory=true`)는 멘토만 수정 가능

과제 삭제

```
DELETE /tasks/{taskId}
```

[멘토] 과제 확인 체크 토글

```
PATCH /mentor/tasks/{taskId}/confirm
```

호출할 때마다 `isMentorChecked`가 `true` ↔ `false` 토글됩니다.

[멘토] 반복 과제 그룹 삭제

```
DELETE /mentor/recurring-tasks/{recurringGroupId}
```

[멘토] 멘티 과제 목록 (날짜별)

```
GET /mentor/tasks/list/{menteeId}?date=2026-02-10
```

[멘티] 내 과제 목록 (날짜별)

```
GET /mentee/tasks/list?date=2026-02-10
```

6. 타이머 (공부 시간)

타이머 시작

```
PATCH /tasks/{taskId}/timer/start
{
  "taskId": 1,
  "timerStatus": "RUNNING",
  "timerStartedAt": "2026-02-05T14:30:00"
}
```

타이머 종료

```
PATCH /tasks/{taskId}/timer/stop
{
  "taskId": 1,
  "sessionSeconds": 1800,
  "sessionFormatted": "00:30:00",
  "accumulatedSeconds": 4500,
  "accumulatedFormatted": "01:15:00"
}
```

- `sessionSeconds` / `sessionFormatted`: 이번 세션 공부시간
- `accumulatedSeconds` / `accumulatedFormatted`: 총 누적 공부시간

누적 공부시간 조회

```
GET /tasks/{taskId}/accumulated-study-time
```

start/stop을 여러 번 반복하면 공부시간이 계속 누적됩니다. 시간은 모두 초 단위로 저장되고, "HH: MM: SS" 포맷 문자열도 함께 응답됩니다.

7. 과제 제출

[멘티] 과제 제출 (이미지 + 메모)

```
POST /tasks/{taskId}/submissions  
Content-Type: multipart/form-data
```

- **files**: 이미지 파일들 (여러 장 가능)
- **comment**: 텍스트 메모 (선택)

제출하면 해당 과제의 status가 자동으로 **DONE**으로 변경됩니다.

제출물 조회

```
GET /tasks/{taskId}/submissions  
{  
    "id": 1,  
    "images": [  
        { "id": 1, "imageUrl": "https://s3.../uid1.jpg" },  
        { "id": 2, "imageUrl": "https://s3.../uid2.jpg" }  
],  
    "menteeComment": "3번 문제 풀이 방법을 모르겠어요.",  
    "createdAt": "2026-02-05T14:30:00",  
    "menteeName": "멘티1"  
}
```

제출물 삭제

```
DELETE /tasks/submissions/{submissionId}
```

8. 피드백

8-1. 플래너 피드백

[멘토] 작성

```
POST /plans/{planId}/feedback
```

```
{ "content": "오늘 계획 잘 세웠어요!" }
```

조회

```
GET /plans/{planId}/feedback
```

[멘토] 수정 / 삭제

```
PUT /plans/{planId}/feedback
```

```
DELETE /plans/{planId}/feedback
```

8-2. 이미지 좌표 피드백

제출된 이미지 위에 좌표를 찍어 피드백을 남기는 기능입니다.

[멘토] 작성

```
POST /images/{imageId}/feedback
```

```
Content-Type: multipart/form-data
```

- **content**: 피드백 텍스트 (<강조>중요한 부분</강조> 태그 사용 가능)
- **xPos**: X 좌표 (0.0 ~ 1.0 비율)
- **yPos**: Y 좌표 (0.0 ~ 1.0 비율)
- **file**: 설명용 이미지 1장 (선택)

imageId는 과제 제출물 조회(GET /tasks/{taskId}/submissions) 응답의 **images[].id** 값입니다.

이미지별 피드백 목록

```
GET /images/{imageId}/feedback
```

```
[  
  {  
    "id": 1,  
    "content": "<강조>공식 적용이 잘못됐어요. </강조> 다시 확인해보세요.",  
    "imageUrl": "https://s3.../feedback.jpg",  
    "xPos": 0.35,
```

```
        "yPos": 0.72,  
        "mentorName": "멘토1",  
        "commentCount": 2,  
        "createdAt": "2026-02-03T14:30:00"  
    }  
]
```

<강조>...</강조> 태그는 프론트에서 파싱해서 밑줄/하이라이트로 렌더링하면 됩니다.

[멘토] 수정 / 삭제

```
PUT /feedback/{feedbackId}  
DELETE /feedback/{feedbackId}
```

8-3. 피드백 목록 조회

[멘티] 어제자 피드백

```
GET /feedbacks/yesterday?page=0&size=10
```

멘티가 피드백 탭에 진입했을 때 사용합니다.

이전 피드백 모아보기 (필터 + 페이지)

```
GET /feedbacks/history?menteeId=2&subject=MATH&year=2026&month=2&page=0&size=10
```

필터 옵션 (모두 선택사항):

- `menteeId` (필수)
- `subject: KOREAN / ENGLISH / MATH / OTHER`
- `year, month`
- `startDate, endDate`

8-4. 피드백 댓글

피드백 아래에 멘토-멘티가 대화하는 댓글 스레드입니다. 텍스트만 가능 (사진 첨부 불가).

댓글 작성

```
POST /feedback/{feedbackId}/comments  
{ "comment": "이 부분 다시 풀어봤는데 맞나요?" }
```

댓글 목록 조회

```
GET /feedback/{feedbackId}/comments
```

```
[  
  {  
    "id": 1,  
    "comment": "이 부분 다시 풀어봤는데 맞나요?",  
    "authorName": "멘티1",  
    "role": "MENTEE",  
    "createdAt": "2026-02-05T15:10:00"  
  },  
  {  
    "id": 2,  
    "comment": "네, 이번에는 정확합니다.",  
    "authorName": "멘토1",  
    "role": "MENTOR",  
    "createdAt": "2026-02-05T16:00:00"  
  }  
]
```

댓글 수정 / 삭제 (본인만)

```
PUT /feedback/comments/{commentId}  
DELETE /feedback/comments/{commentId}
```

9. 줌 미팅 피드백

[멘토] 작성

```
POST /mentor/zoom-feedback/{menteeId}  
{  
  "title": "6월 1주차 줌 미팅",  
  "meetingDate": "2026-06-03",  
  "memo": "전체적으로 좋아짐",  
  "koreanFeedback": "비문학 독해 속도 향상",  
  "englishFeedback": "문법 보완 필요",  
  "mathFeedback": "계산 실수 주의",  
  "operateFeedback": "플래너 작성 습관 좋음"  
}
```

[멘토] 상세 조회 / 수정 / 삭제

```
GET /mentor/zoom-feedback/{feedbackId}  
PUT /zoom-feedback/{feedbackId}  
DELETE /mentor/zoom-feedback/{feedbackId}
```

[멘토] 멘티별 목록

```
GET /mentor/list/{menteeId}?page=0&size=20
```

[멘티] 내 좀 피드백 목록 / 상세

```
GET /mentee/list?page=0&size=20  
GET /mentee/zoom-feedback/{feedbackId}
```

10. 보완점 (약점)

[멘토] 생성

```
POST /mentor/weakness
```

```
{  
    "title": "이차방정식 풀이 미숙",  
    "menteeId": 2,  
    "subject": "MATH",  
    "contentId": 5  
}
```

- **contentId**: 연결할 학습지 ID (선택)

[멘토] 멘티별 조회

```
GET /mentor/weakness/{menteeId}?subject=MATH&page=0&size=20
```

[멘티] 내 보완점 조회

```
GET /mentee/weakness/me?subject=MATH&page=0&size=20
```

[멘토] 삭제

```
DELETE /mentor/weakness/{weaknessId}
```

11. 학습지 (PDF)

[멘토] 업로드

```
POST /study-contents  
Content-Type: multipart/form-data
```

- **title**: 학습지 제목
- **subject**: KOREAN / ENGLISH / MATH

- file: PDF 파일

목록 조회

```
GET /study-contents?page=0&size=20
```

[멘토] 삭제

```
DELETE /study-contents/{contentId}
```

12. 주간 학습 리포트

[멘토] 작성

```
POST /mentor/weekly-report
```

```
{  
    "menteeId": 2,  
    "reportYear": 2026,  
    "reportMonth": 6,  
    "weekNumber": 1,  
    "startDate": "2026-06-01",  
    "endDate": "2026-06-07",  
    "overallFeedback": "공부 습관이 잡혀가고 있어요.",  
    "strengths": "수학 정확도 향상",  
    "weaknesses": "영어 독해 시간 부족"  
}
```

목록 조회 (필터)

```
GET /weekly-reports?menteeId=2&year=2026&month=6&page=0&size=20
```

상세 조회

```
GET /weekly-reports/{reportId}
```

[멘토] 수정 / 삭제

```
PATCH /mentor/weekly-report/{reportId}  
DELETE /mentor/weekly-report/{reportId}
```

13. 알림

FCM 푸시 알림 연동 (React + Vite)

1단계: Firebase 설치

```
npm install firebase
```

2단계: Firebase 초기화 ([src/firebase.ts](#))

```
import { initializeApp } from 'firebase/app';
import { getMessaging, getToken, onMessage, type MessagePayload } from
'firebase/messaging';

const firebaseConfig = {
  apiKey: "Firebase 콘솔에서 복사",
  authDomain: "프로젝트ID.firebaseio.com",
  projectId: "프로젝트ID",
  messagingSenderId: "발신자ID",
  appId: "앱ID",
};

const app = initializeApp(firebaseConfig);
export const messaging = getMessaging(app);

// FCM 토큰 발급
export async function requestFcmToken(): Promise<string | null> {
  const permission = await Notification.requestPermission();
  if (permission !== 'granted') return null;

  const token = await getToken(messaging, {
    vapidKey: 'Firebase 콘솔 > 클라우드 메시징 > 웹 푸시 인증서 키',
  });
  return token;
}

// 포그라운드 메시지 수신 (앱이 열려있을 때)
export function onForegroundMessage(callback: (payload: MessagePayload) => void) {
  onMessage(messaging, (payload) => {
    callback(payload);
  });
}
```

[firebaseConfig](#)와 [vapidKey](#)는 [Firebase 콘솔](#) > 프로젝트 설정에서 확인할 수 있습니다.

3단계: Service Worker 파일 (`public/firebase-messaging-sw.js`)

```
// Service Worker는 TypeScript가 아닌 JavaScript로 작성합니다
importScripts('https://www.gstatic.com/firebasejs/9.0.0.firebaseio-compat.js');
importScripts('https://www.gstatic.com/firebasejs/9.0.0/firebase-messaging-
compat.js');

firebase.initializeApp({
  apiKey: "위와 동일",
  authDomain: "위와 동일",
  projectId: "위와 동일",
  messagingSenderId: "위와 동일",
  appId: "위와 동일",
});

const messaging = firebase.messaging();
```

// 백그라운드 메시지 수신 (앱이 닫혀있을 때)

```
messaging.setBackgroundMessage((payload) => {
  const { title, body } = payload.notification;
  self.registration.showNotification(title, { body });
});
```

4단계: 로그인 후 백엔드에 토큰 등록

```
import { requestFcmToken } from './firebase';
import axios from 'axios';

// 로그인 성공 후 호출
async function registerFcmToken(): Promise<void> {
  const fcmToken = await requestFcmToken();
  if (fcmToken) {
    await axios.post('/users/me/fcm-token', { fcmToken });
  }
}
```

5단계: 포그라운드 알림 표시 (선택)

```
import { onForegroundMessage } from './firebase';

// App.tsx 등에서 한 번 호출
onForegroundMessage((payload) => {
  // toast, 알림 뱃지 등 원하는 방식으로 표시
  alert(payload.notification?.body);
});
```

요약: Firebase SDK로 토큰 발급 → `POST /users/me/fcm-token`으로 백엔드에 등록 → 이후 서버가 알아서 푸시 전송

알림 목록 조회

```
GET /notifications?filter=unread&page=0&size=20
```

- filter: all (전체) / unread (안읽음) / read (읽음)

알림 읽음 처리

```
PATCH /notifications/{notificationId}/read
```

전체 읽음 처리

```
PATCH /notifications/read-all
```

알림 자동 발생 트리거

아래 동작을 하면 서버가 자동으로 알림을 생성하고, FCM 토큰이 등록된 사용자에게 푸시를 보냅니다.

트리거	수신자	알림 메시지 예시
멘토가 과제 출제	멘티	"김멘토님이 새 과제를 출제했습니다: 수학 이차방정식"
멘티가 과제 제출	멘토	"최연준 멘티가 과제를 제출했습니다: 수학 이차방정식"
멘토가 이미지 좌표 피드백 작성	멘티	"김멘토님이 피드백을 남겼습니다."
멘토가 플래너 피드백 작성	멘티	"김멘토님이 플래너 피드백을 작성했습니다."
멘토/멘티가 피드백 댓글 작성	상대방	"김멘토님이 댓글을 남겼습니다."
멘토가 줌 피드백 작성	멘티	"김멘토님이 줌 미팅 피드백을 작성했습니다."
멘토가 주간 리포트 작성	멘티	"김멘토님이 주간 리포트를 작성했습니다."
매일 자정 (자동)	미완료 과제가 있는 멘티	"오늘 완료하지 않은 과제가 3개 있습니다."

프론트에서 별도로 알림 생성 API를 호출할 필요 없습니다. 위 동작을 하면 서버가 알아서 처리합니다. 푸시를 받으려면 로그인 후 `POST /users/me/fcm-token`으로 FCM 토큰을 등록해야 합니다.

14. 대시보드 (마이페이지)

[멘토] 멘티 대시보드 조회

```
GET /mentor/{menteeId}?type=WEEK
```

[멘티] 내 대시보드 조회

```
GET /mentee/me?type=WEEK
```

- `type: WEEK` (1주) / `MONTH` (1개월)

응답:

```
{  
    "menteeId": 2,  
    "name": "멘티1",  
    "profileImgUrl": "https://... ",  
    "schoolName": "00고등학교",  
    "submittedTasksCount": 3,  
    "remainingTasksCount": 5,  
    "feedbackQuestionsCount": 2,  
    "todayPlannerTodoCount": 1,  
    "koreanProgress": 72.0,  
    "mathProgress": 47.0,  
    "englishProgress": 100.0  
}
```

필드 설명:

- `submittedTasksCount`: 제출했지만 멘토가 아직 확인 안 한 과제 수
 - `remainingTasksCount`: 미완료 필수 과제 수
 - `feedbackQuestionsCount`: 멘토가 안 읽은 멘티 댓글 수
 - `todayPlannerTodoCount`: 오늘 플래너에 멘토 피드백이 없으면 1
 - `koreanProgress` / `mathProgress` / `englishProgress`: 과목별 진행률 (%)
-

에러 응답 형식

모든 에러는 아래 형식으로 응답됩니다:

```
{  
  "code": "USER_NOT_FOUND",  
  "message": "존재하지 않는 사용자입니다."  
}
```

주요 에러 코드

코드	HTTP	설명
USER_NOT_FOUND	404	사용자 없음
UNAUTHORIZED_ACCESS	403	권한 없음
PLAN_NOT_FOUND	404	플래너 없음
TASK_NOT_FOUND	404	과제 없음
TASK_NOT_MODIFIABLE	403	과제 수정 불가 (필수 과제를 멘티가 수정 시도 등)
TIMER_ALREADY_RUNNING	400	이미 타이머 실행 중
TIMER_NOT_RUNNING	400	타이머가 실행 중이 아님
PLAN_DUPLICATE_DATE	409	같은 날짜에 플래너 이미 존재
MENTEE_INFO_NOT_FOUND	404	멘토-멘티 매핑 없음

과목(Subject) Enum 값

값	의미
KOREAN	국어
ENGLISH	영어
MATH	수학
OTHER	기타

페이지 파라미터

페이지가 있는 API는 공통으로 아래 파라미터를 사용합니다:

- **page**: 페이지 번호 (0부터 시작)
- **size**: 한 페이지 크기 (기본값 20)

응답에 포함되는 페이지 정보:

```
{  
    "content": [...],  
    "page": 0,  
    "size": 20,  
    "total Elements": 42,  
    "total Pages": 3  
}
```