**[매칭 기능]**

**\*\* 매칭 시스템 \*\***

* 매일 전역적으로 알고리즘이 돌아가서 랜덤하게 매칭 해준다.
* DB의 카운트만큼 배열의 인덱스에 각각 인덱스 값을 할당하고 랜덤하게 섞기. (중복 매칭 및 자기 자신과의 매칭 알아서 피해진다.)
* 두 개씩 짝지어서 ‘ChatSessions’에 업데이트하기

**- 전체 코드**

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- 이 코드는 Node.js와 Express를 사용하여 작성된 매칭 시스템의 일부로, 주기적으로 유저를 매칭하고 매칭 상태를 관리하는 API 엔드포인트들을 제공한다.**

**[주요 필드 및 함수]**

**<주요 필드>**

- **User 모델과 ChatSessions 모델**

: ‘User’와 ‘ChatSessions’는 MongoDB 컬렉션을 참조하는 Mongoose 모델로 가정된다.

**<주요 함수>**

- ‘**cron.schedule(‘0 0 \* \* \*’, async () => {…}:**’

* 매일 자정에 실행되는 cron 작업을 정의한다.
* 미칭되지 않은 유저들을 찾고, 이를 랜덤하게 섞어 매칭한다.
* 매칭된 유저의 상태를 업데이트하고 새로운 ‘ChatSessions’를 생성한다.
* 서버를 재시작한다.

- **‘app.post('/api/checkmatch', async (req, res) => { ... }):’**

* 특정 유저의 매칭 상태를 확인하는 API 엔드포인트이다.
* 클라이언트로부터 유저 ID를 받아 해당 유저의 매칭 상태를 반환한다.

- **‘app.put('/api/choosematch', async (req, res) => { ... }):’**

* 유저가 매칭을 수락 또는 거절했을 때 호출되는 API 엔드포인트이다.
* 클라이언트로부터 유저 ID와 매칭 상태를 받아 처리한다.
* 상대방 유저의 매칭 상태에 따라 최종 매칭 결과를 결정하고 업데이트한다.

**<코드 분석>**

**1. cron.schedule**

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명

- **주요 작업**

* **유저 검색**: 매칭되지 않은 유저들을 데이터베이스에서 검색한다.
* **랜덤 섞기**: 검색된 유저들을 랜덤하게 섞습니다. 이를 통해 매칭의 무작위성을 보장한다.
* **매칭 및 상태 업데이트**: 유저들을 2명씩 짝지어 새로운 채팅 세션을 생성하고, 매칭 상태를 업데이트한다.
* **서버 재시작**: 모든 작업이 끝난 후, 서버를 재시작 하여 변경사항을 반영한다.

**- usersToMatch**

: 매칭되지 않은 유저들의 목록을 저장하는 배열.

**- usersCount**

: 매칭되지 않은 유저의 수

**- arrayForShuffle**

: 유저 인덱스를 섞기 위한 배열

**- userIds**

: 매칭될 유저의 ID를 저장하는 배열.

**<세부 설명>**

**1. 매칭되지 않은 유저 찾기.**



* ‘Users’ 컬렉션에서 ‘matchState’가 'matched'가 아닌 모든 유저를 찾는다.
* ‘usersToMatch‘배열에 결과를 저장하고, 유저 수를 ‘usersCount’에 저장한다.

**2. 유저 인덱스를 섞기 위한 배열 생성 및 셔플**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* ‘arrayforShuffle’ 배열을 생성하고, 유저 인덱스(0부터 usersCount-1까지)를 할당한다.
* Fisher-Yates 셔플 알고리즘을 사용하여 배열을 무작위로 섞는다.

**3. 유저 매칭 및 상태 업데이트**

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* 두 명씩 짝을 지어 매칭한다.
* ChatSessions 컬렉션에 새로운 채팅 세션을 생성한다.
* 각 유저의 matchState를 'choose'로 설정하고, 이를 데이터베이스에 저장한다.

**4. 서버 재시작**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* ‘exec’ 함수를 사용하여 서버를 재시작 한다.
* 재시작 과정에서 오류가 발생하면 로그에 기록한다.

**2. app.post(‘/api/checkmatch’)**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- 관련 필드**

: **‘userId’**: 클라이언트로부터 전달된 유저 ID

**- 주요 작업**

* **유저 매칭 상태 확인**: 클라이언트로부터 전달된 유저 ID를 바탕으로 데이터베이스에서 해당 유저를 찾고 매칭 상태를 반환한다.

**<세부 설명>**

- 이 API 엔드포인트는 특정 유저의 매칭 상태를 확인한다.

**1. 유저의 ID 추철 및 유저 검색**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* 클라이언트 요청의 본문에서 userId를 추출한다.
* 데이터베이스에서 해당 유저를 검색한다.
* 유저의 matchState를 클라이언트에 JSON 형태로 반환한다.
* 검색 도중 오류가 발생하면 로그에 기록하고, 500 상태 코드를 반환한다.

**3. app.put('/api/choosematch')**

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- 관련 필드**

* **userId**: 클라이언트로부터 전달된 유저 ID.
* **matchState**: 클라이언트로부터 전달된 매칭 상태.
* **chatSession**: 현재 유저가 속한 채팅 세션.
* **matchUserId**: 매칭된 상대방 유저 ID.
* **matchUser**: 매칭된 상대방 유저.

**- 주요 작업**

* **유저 상태 업데이트**: 클라이언트로부터 전달된 유저 ID와 매칭 상태를 바탕으로, 유저와 매칭 상대방의 매칭 상태를 업데이트한다.
* **상태별 처리**

**🡪** ‘**pending**’: 상대방도 ‘pending’ 상태이면 두 유저를 ‘matched’로 설정한다.

**🡪** ‘**notMatched**’: 두 유저를 ‘notMatched’로 설정한다.

**<세부 설명>**

이 API 엔드포인트는 유저가 매칭을 수락 또는 거절했을 때 호출된다.

**1. 유저 ID 및 메칭 상대 추출**



* **유저 검색**: 전달된 ‘userId’를 사용하여 현재 유저와 매칭된 상대방 유저를 검색한다.

**2. 현재 유저와 매칭 상대방 유저 검색**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명



* **상대방 유저 상태 확인**: 상대방 유저의 ‘matchState’가 ’notMatched'인 경우, 현재 유저도 'notMatched'로 설정하고 종료한다.

**3. 매칭 상태 업데이트**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

* **현재 유저 상태 업데이트**: 현재 유저의 ‘matchState’를 클라이언트가 보낸 값으로 설정한다.
* **양쪽 매칭 확인**: 현재 유저의 상태가 'pending'인 경우, 상대방 유저도 'pending'인지 확인합니다. 둘 다 'pending'이면 'matched'로 설정한다.
* **매칭 거절 처리**: 현재 유저가 매칭을 거절한 경우, 두 유저의 상태를 ‘notMatched’로 설정한다.

**4. 오류 처리**

**텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명**

위의 코드를 이해하기 위해 알아야 할 기본 지식을 정리하겠습니다.

- **Cron 작업**

: Cron 작업은 리눅스에서 주기적으로 특정 작업을 수행하기 위해 사용되는 스케줄러이다.

🡪 Node.js에서는 ‘node-cron’ 패키지를 사용하여 cron 작업을 정의할 수 있다.

**- MongoDB와 Mongoose**

: MongoDB는 NoSQL 데이터베이스로, 유연한 데이터 모델을 제공한다.

* **Mongoose**: MongoDB와 상호작용하는 ODM(Object Document Mapper) 라이브러리로, 스키마 정의, 데이터 검증, 쿼리 빌딩 등을 지원한다.

**- 비동기 처리**

: ‘async’와 ‘await’를 사용하여 비동기 코드를 작성할 수 있다.

🡪 코드의 가독성을 높이고, 비동기 작업의 결과를 처리할 때 편리하다.

**- Express**

: Express는 Node.js를 위한 간결하고 유연한 웹 프레임워크이다.

🡪 API 서버를 구축할 때 자주 사용되며, 다양한 HTTP 메소드와 미들웨어를 지원한다.

**- 오류 처리**

: 비동기 함수 내에서 발생하는 오류를 catch 블록에서 처리하여, 서버의 안정성을 높인다.

🡪 try-catch 블록을 사용하여 수행된다.

**<코드 분석 요약>**

* 이 코드는 유저 매칭 시스템을 구현하는 Node.js와 Express 기반의 API이다.
* 매일 자정에 cron 작업을 통해 매칭되지 않은 유저를 랜덤하게 매칭하고, 두 개의 API 엔드포인트 (/api/checkmatch, /api/choosematch)를 통해 유저의 매칭 상태를 확인 및 업데이트한다.
* MongoDB와 Mongoose를 사용하여 데이터베이스 작업을 수행하며, 비동기 처리를 통해 효율적으로 작업을 처리한다.