```
import { CREATE_MESSAGE_REQUEST, READ_MESSAGE_REQUEST,
MESSAGE_CREATED, UPDATED_UNREAD_MESSAGE_COUNT,
READ_MESSAGE_CONFIRMED, MATCH_REQUEST,
ALL_USERS_MATCHING_CHANNEL, MATCH_NOT_FOUND, REFRESH_USER,
MATCH_FOUND, READ_MATCH_MESSAGES_CONFIRMED,
READ_MATCH_MESSAGES_REQUEST } from "../../constants/socket-
events/index.js";
import { FULFILLED, UNFULFILLED } from "../../constants/match-
statuses/index.js";

import Match from "../../db/models/Match.js";
import Message from "../../db/models/Message.js";
import { addPostCommitFn, commitTrx, rollbackTrx, startTrx }
from "../../utils/transaction-utils/index.js";
import * as Ably from "ably/promises.js";
import messageCrud from "../../api-services/message/
messageCrud.js";
import userCrud from "../../api-services/user/userCrud.js";
import matchCrud from '../../api-services/match/matchCrud.js';
import questionSubscriptionCrud from '../../api-services/
question-subscription/questionSubscriptionCrud.js';

import { log } from 'next-axiom'

import { fileURLToPath } from "url";

const filename = fileURLToPath(import.meta.url);

class AblyService {
    ably;

    initialize = async (passedTrx = null) => {
        let trx;

        try {
            log.debug('ABLY INITIALIZE', { testData: 32423 })

            const realtime = new Ably.Realtime({
                key: process.env.VERCEL_ENV === "production" ?
process.env.ABLY_API_KEY_PRODUCTION :
process.env.ABLY_API_KEY_DEVELOPMENT,
                log: { level: 4 },
            });

            this.ably = realtime;
```

```
            this.ably.connection.once("connected");
            console.log("NOW Connected to Ably!!");

            trx = await startTrx(Match, passedTrx);

            await this.setUpAllUsersChannelSubscriptions({});

            const allMatches = await Match.query(trx);

            for (const match of allMatches) {
                await this.setUpChannelSubscriptions({ match_id:
match.match_id, passedTrx: trx });
            }

            await commitTrx(trx, passedTrx);
        } catch (err) {
            console.error({
                filename,
                function: "initialize",
                message: `Failed to initialize ably service: $
{err}`,
            });

            await rollbackTrx(trx, err);

            throw err;
        }
    }

    setUpAllUsersChannelSubscriptions = async ({ passedTrx =
null }) => {
        let trx;

        try {
            trx = await startTrx(Match, passedTrx);

            console.log("in
setUpAllUsersChannelSubscriptions!!!")
            log.debug('In set up all users challen
subscriptions')

            // get the finding match channel
            const allUsersMatchChannel =
this.ably.channels.get(ALL_USERS_MATCHING_CHANNEL);
```

```
            log.debug('Found all users match channel!!!',
{ allUsersMatchChannel: !!allUsersMatchChannel })

            await allUsersMatchChannel.subscribe(MATCH_REQUEST,
async (message) => {
                console.log("GOT MATCH REQUEST!!! message: ",
message)
                log.debug('Listened to MATCH_REQUEST',
{ message: !!message })
                const currentUserChannel =
this.ably.channels.get(`user-${message.data.user_id}`);


                await this.mockOnMatchRequest({
                    user_id: +message.data.user_id,
                    question_id: +message.data.question_id,
                    currentUserChannel: currentUserChannel,
                });
            })

            await commitTrx(trx, passedTrx);

        } catch (err) {
            console.error({
                filename,
                function: "setUpAllUsersChannelSubscriptions",
                message: `Failed to set up finding match channel
subscriptions: ${err}`,
            });

            await rollbackTrx(trx, err);

            throw err;
        }
    }


    mockOnMatchRequest = async ({ user_id, question_id,
currentUserChannel, passedTrx = null }) => {
        console.log('mockOnMatchRequest! SENDING MATCH NOT
FOUND')

        const leftUserChannel = this.ably.channels.get(`match-
finding-user-${user_id}`);
```

```
        console.log('Sending to left user channel',
{ leftUserChannel: !!leftUserChannel })

        await leftUserChannel.publish(MATCH_NOT_FOUND, {});
    }

}

const ablyService = new AblyService();

export default ablyService;
```