C🍪🍪KIE-APP

Convergent Capstone Design 2
Week 4 - Progress Report

Group 6 Park JongBeum & Baek SeungHeon, September 25th 2023

Recall

iPhone 14 Pro Max
iOS 16.4

3:33

DEBUG

친구

박종범
백승헌
김채원
홍은채
사쿠라
허윤진
카즈하
Jane
JB
cw4
cw5
cw4

친구

iPhone 14 Pro Max
iOS 16.4

3:35

DEBUG

채팅

박종범 19:23 PM
채팅탭에서 보이는 메시지입니다. 48

김채원 19:23 PM
채팅탭에서 보이는 메시지입니다. 48

백승헌 19:23 PM
채팅탭에서 보이는 메시지입니다. 48

홍은채 19:23 PM
채팅탭에서 보이는 메시지입니다. 48

카즈하 19:23 PM
채팅탭에서 보이는 메시지입니다. 48

사쿠라 19:23 PM
채팅탭에서 보이는 메시지입니다. 48

허윤진 19:23 PM
채팅탭에서 보이는 메시지입니다. 48

김채원 19:23 PM
채팅탭에서 보이는 메시지입니다. 48

김채원 19:23 PM
48

채팅

iPhone 14 Pro Max
iOS 16.4

3:25

COOKIE

김채원 33
안녕하세요!안녕하세요!안녕하세요!안녕하세요!안녕하세요!안녕하세요!

채팅하기
친구신청

C🍪🍪KIE-APP

Frontend

- Sign In & Sign Up
- Friends Tab
- Chat Tab & Chat Room
- Maps Tab & Settings Tab
- MVVM Structure
- State Management
- Error Handling

C🍪🍪KIE-APP

Backend

- RESTful API
- Authorization & Authentication API
- Account Info API
- DB
- (socket.io)

Progress

Maps Tab

Maps Tab
Location permissions

- We need location information in the Background as well

```dart
Future<bool> _checkLocationPermission() async {
  final access = await LocationPermissions().checkPermissionStatus();
  switch (access) {
    case PermissionStatus.unknown:
    case PermissionStatus.denied:
    case PermissionStatus.restricted:
      final permission = await LocationPermissions().requestPermissions(
        permissionLevel: LocationPermissionLevel.locationAlways,
      );
```

U+ 9:57 67%

위치 액세스 권한

cookie_app

이 앱의 위치 액세스 권한

- (●) 항상 허용
- ( ) 앱 사용 중에만 허용
- ( ) 항상 확인
- ( ) 허용 안함

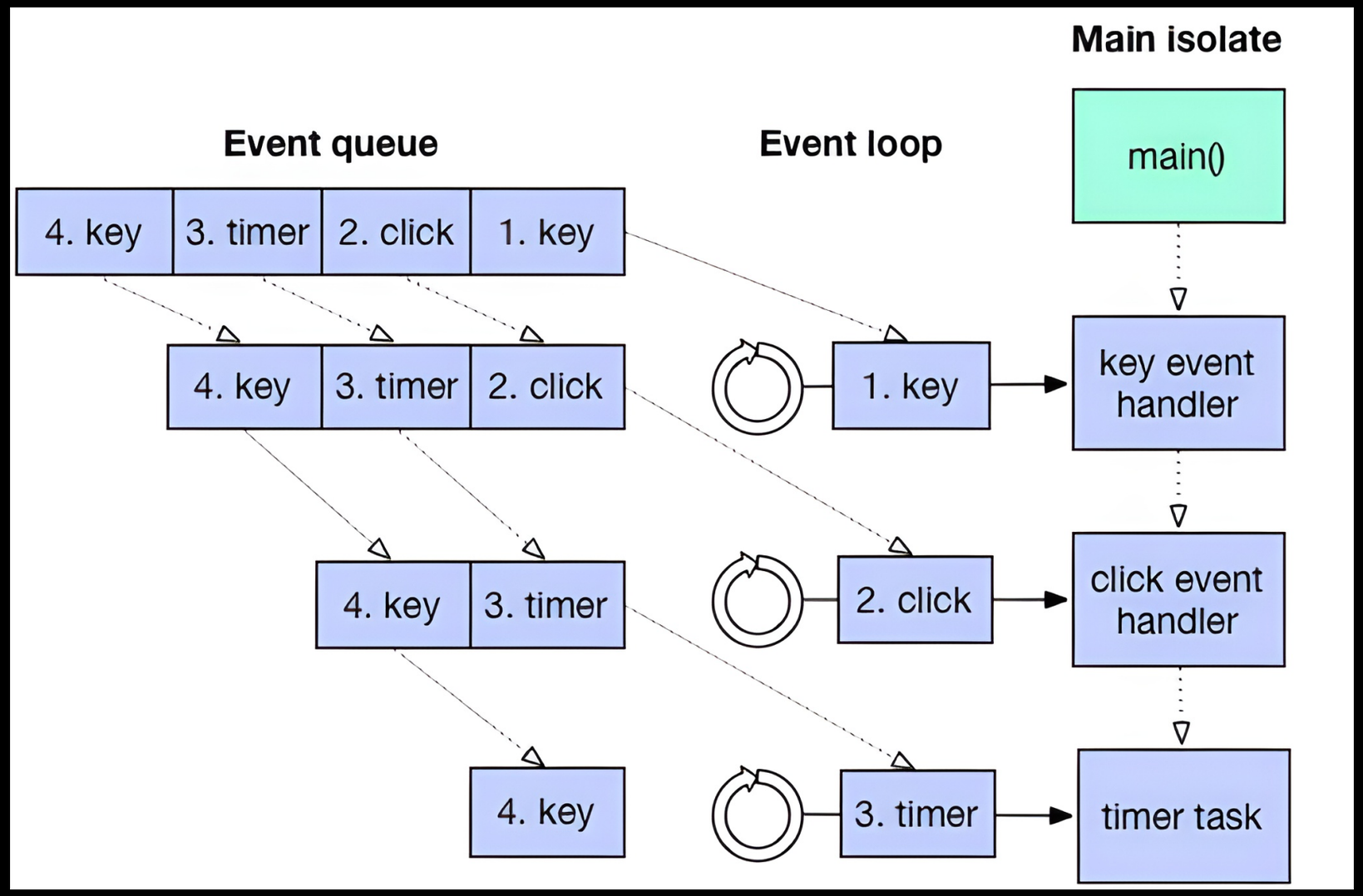
정확한 위치 사용

정확한 위치가 사용 중지된 경우 앱이 대략적인 위치 정보에 액세스할 수 있습니다.

Maps Tab

How to get the location in the background

- Dart is based on a single thread (or isolate)



Maps Tab

How to get the location in the background

```
I/Choreographer( 4682): Skipped 35 frames!  The application may b
e doing too much work on its main thread.
I/Choreographer( 4682): Skipped 80 frames!  The application may b
e doing too much work on its main thread.
```

Maps Tab

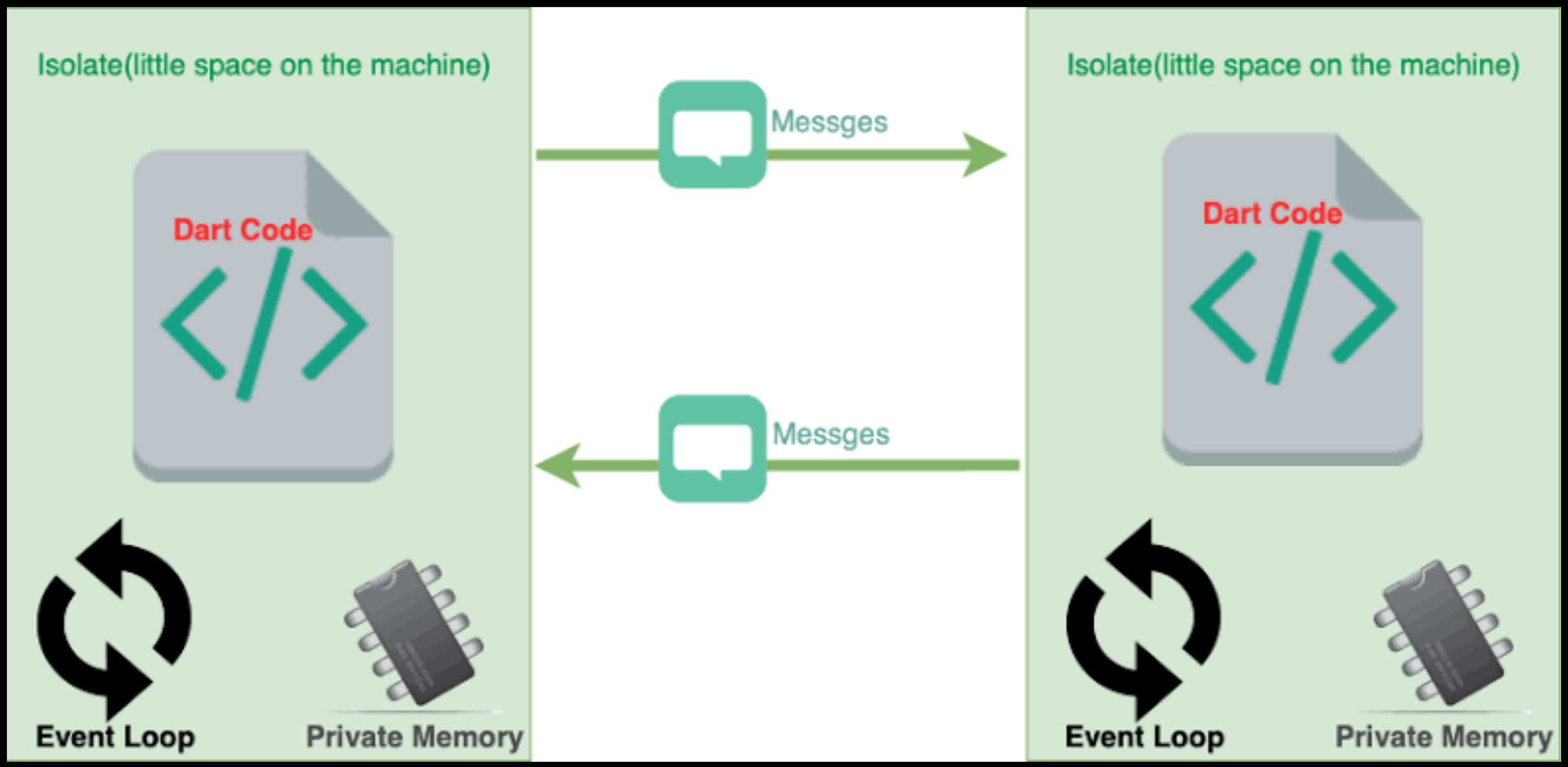
How to get the location in the background

- We need to know the location information continuously, even in the background

- So we use the additional "isolate"

- Create a separate memory space for managing location information

Isolate


Isolate(little space on the machine)
Dart Code
Event Loop
Private Memory
Messges
Messges
Isolate(little space on the machine)
Dart Code
Event Loop
Private Memory

Isolate
send

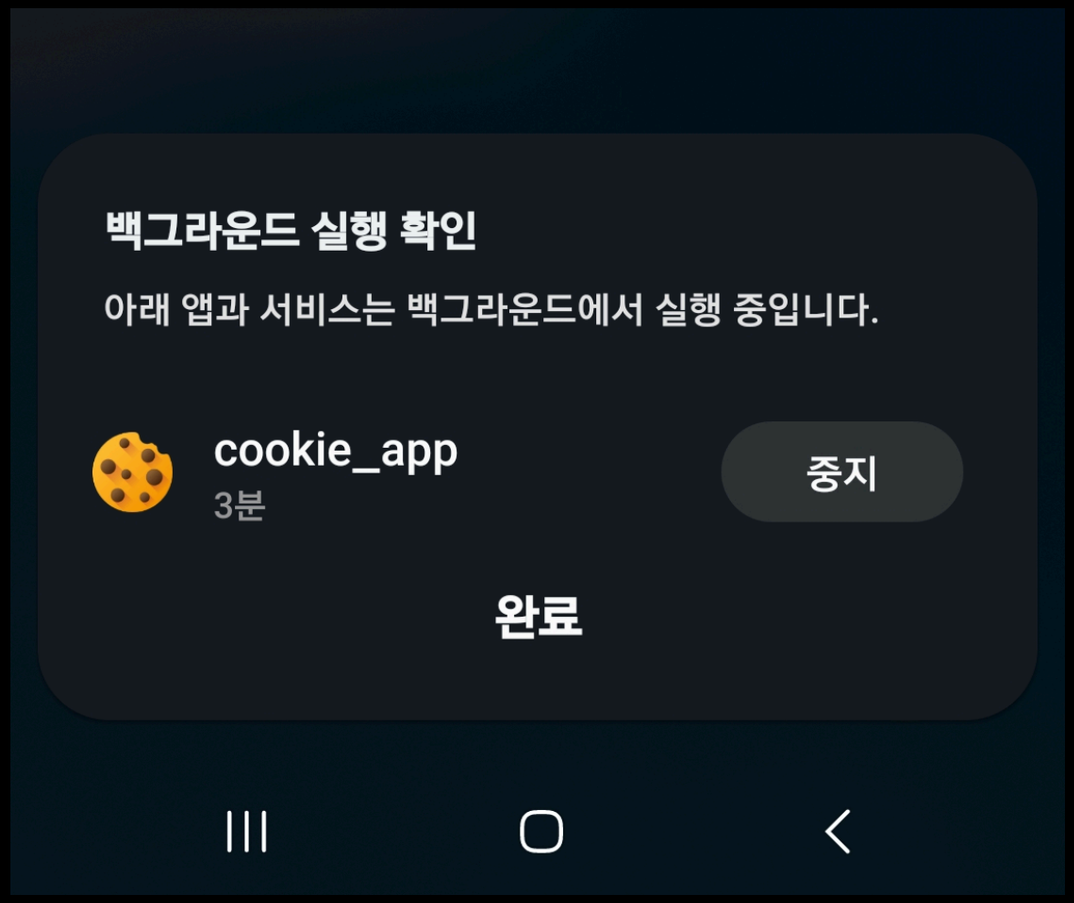
```dart
final SendPort? send =
IsolateNameServer.lookupPortByName(isolateName);

send?.send(locationDto.toJson());
```

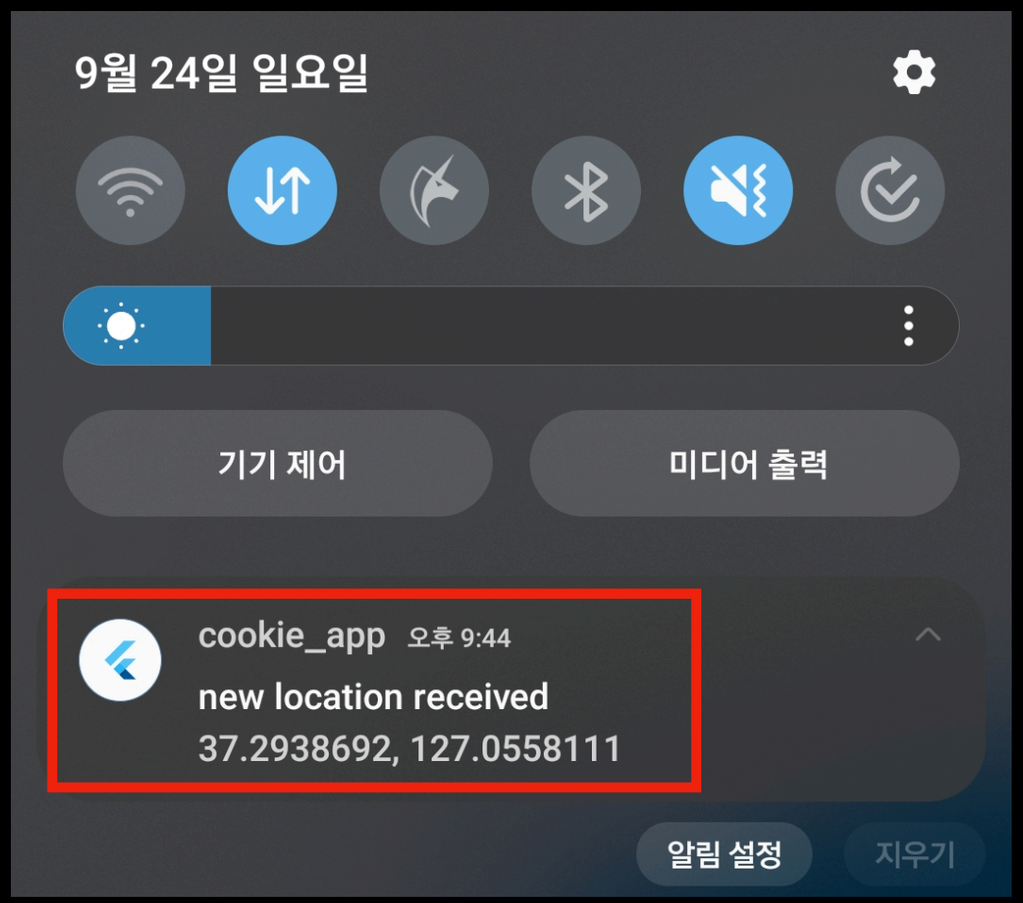
Isolate
receive

```dart
ReceivePort port = ReceivePort();
```

```dart
port.listen(
  (dynamic data) async {
    await update(data);
  },
);
```

Run app in background





Firebase Cloud Messaging

9:18 Sun, Sep 10 100%

Internet

Bluetooth

Flashlight

Do Not Disturb

cookie_app • now

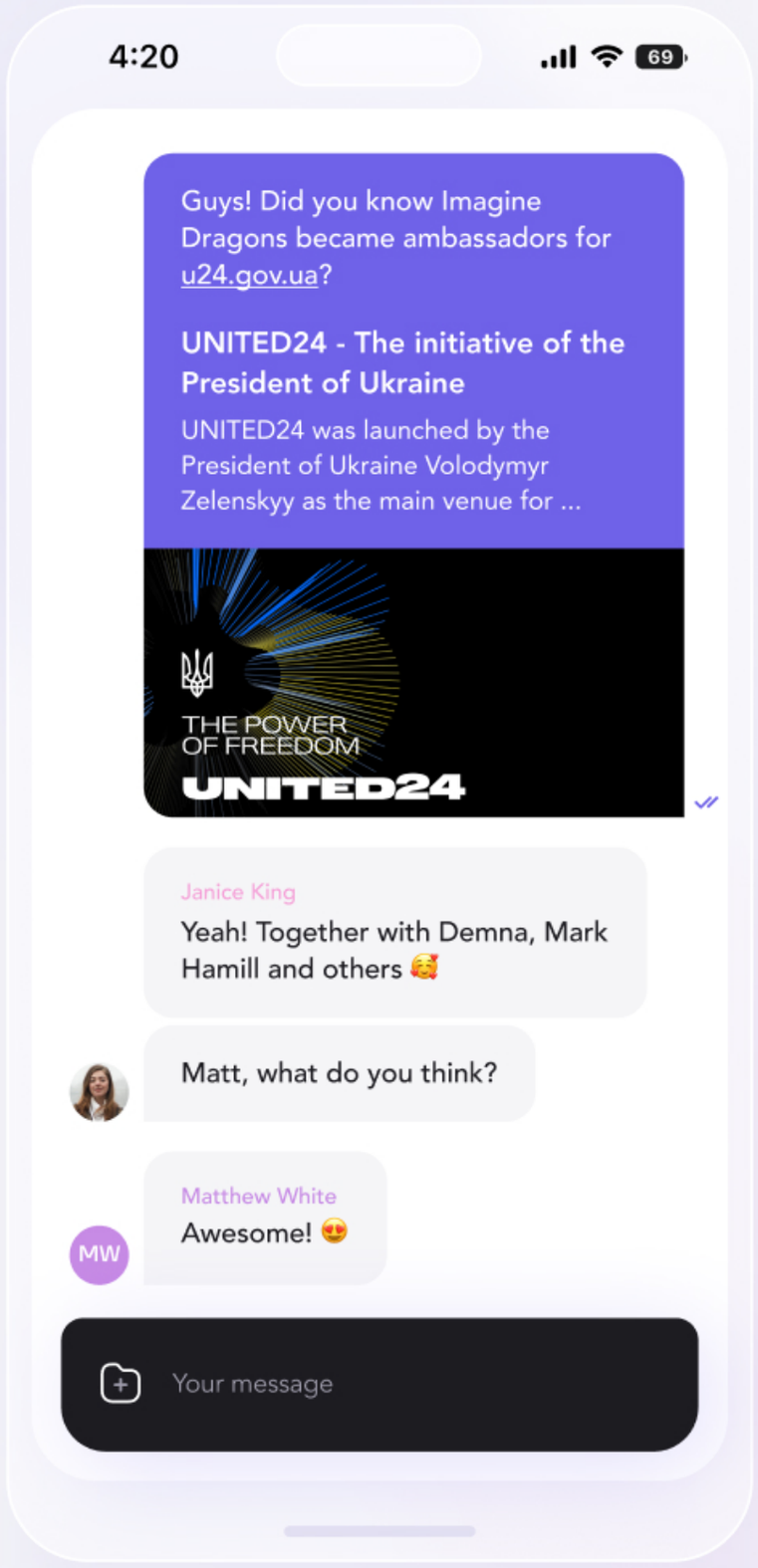
test

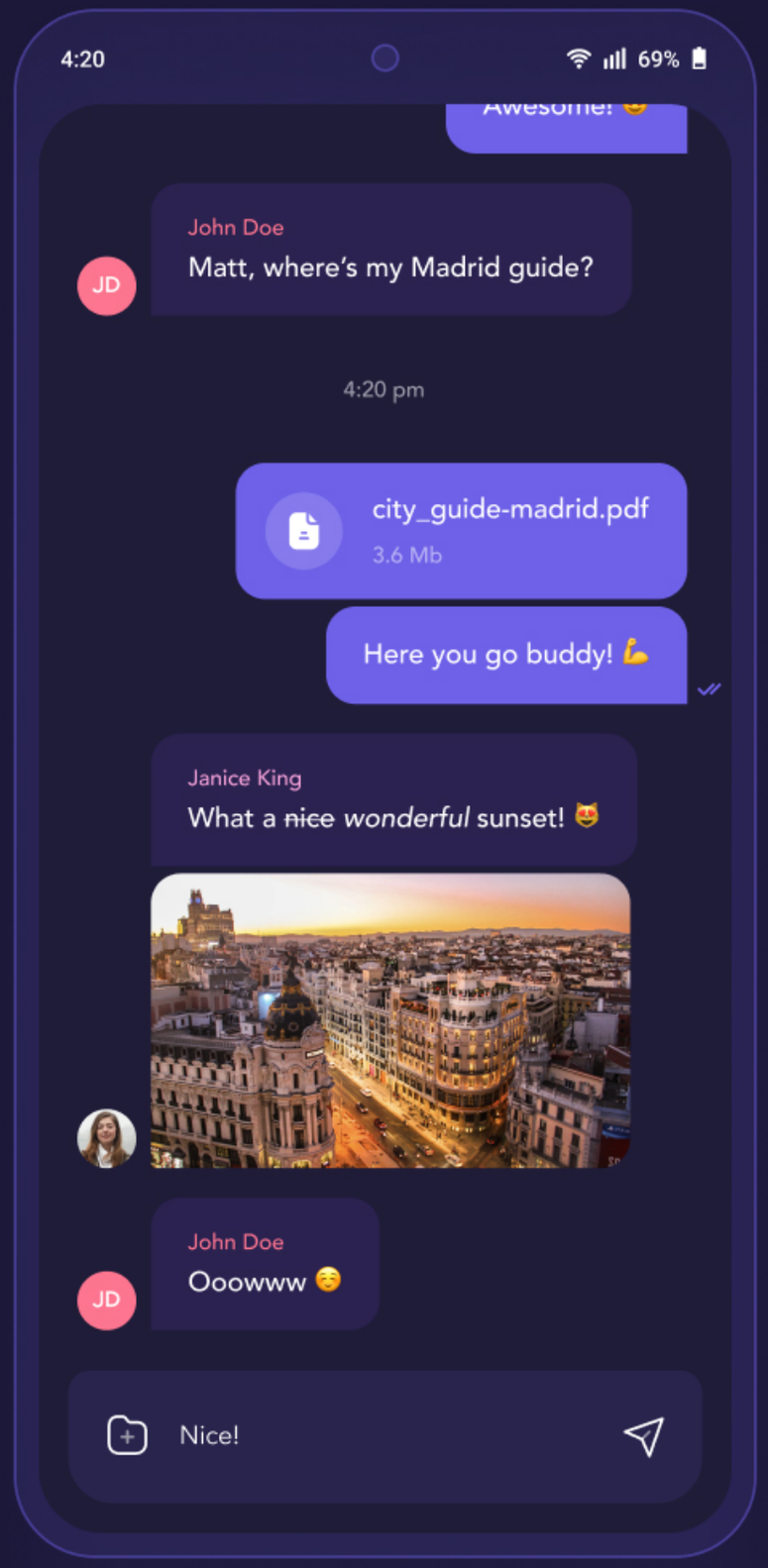
this is test notification

Manage

Clear all

Chatroom Updates


4:20
69
Guys! Did you know Imagine Dragons became ambassadors for u24.gov.ua?
UNITED24 - The initiative of the President of Ukraine
UNITED24 was launched by the President of Ukraine Volodymyr Zelenskyy as the main venue for ...
THE POWER OF FREEDOM
UNITED24
Janice King
Yeah! Together with Demna, Mark Hamill and others 🥰
Matt, what do you think?
Matthew White
Awesome! 😍
MW
Your message


4:20
69%
Awesome! 😍
John Doe
Matt, where's my Madrid guide?
JD
4:20 pm
city_guide-madrid.pdf
3.6 Mb
Here you go buddy! 💪
Janice King
What a ~~nice~~ *wonderful* sunset! 😍
John Doe
Ooowww 😉
JD
Nice!

iPhone 15 Pro Max
iOS 17.0

8:38

채팅

20:36

Test Message Here

Hello World

mongoosejs.com

Mongoose ODM v7.5.2

Message

server

on

new_room. (room{ }, users)

join_room (roomId, users)

leave_room. (roomId).

chat. (roomId, message)

emit.

includes client. emitter

new_room. (room info) → 참여한 users

join_room (roomId, user) → 참여한 user + room내의 users.

leave_room. (roomId, user). → room내의 users

chat. (roomId, from, message) → room내의 users

rooms{}

- id.
- name.
- users.
- messages.

message

- content.
- from.
- time.

client.

emit.

c.e. new_room. 새 톡방 생성.

join_room. 톡방에 초대.

leave_room 톡방에서 퇴장.

chat. 채팅.

on.

new_room.

c.e 이면 :

else 면 : 새 톡방표시.

join_room.

기존 room에 입장한 사람 : 새 톡방표시.

기존 room에 있던 사람들. : UI표시.

leave_room (roomId_user).

c.e 이면. 나가고 톡방삭제.

else 면 UI표시.

chat.

c.e 이면. 내 말풍선.

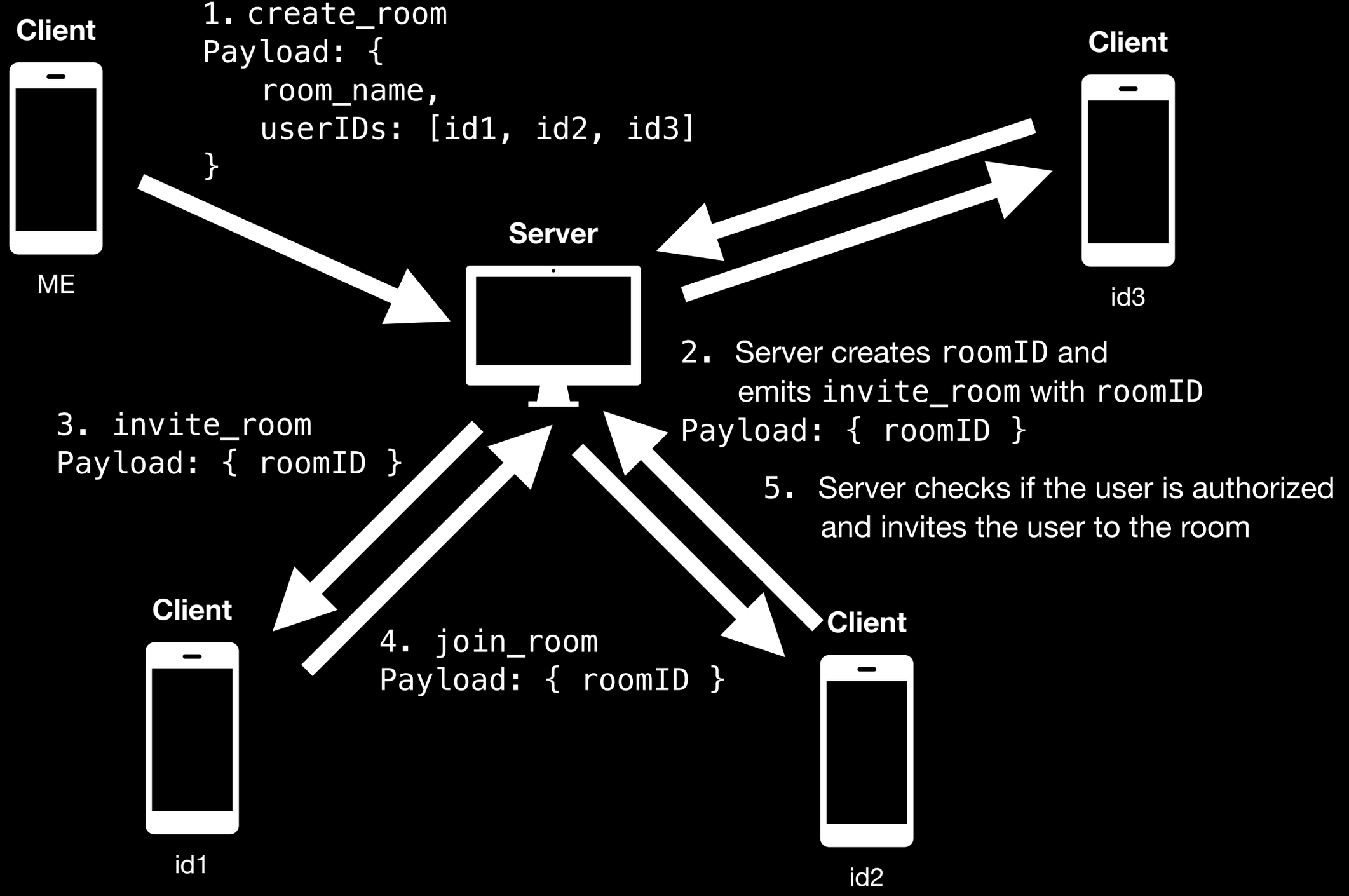
else 면 남의 말풍선.

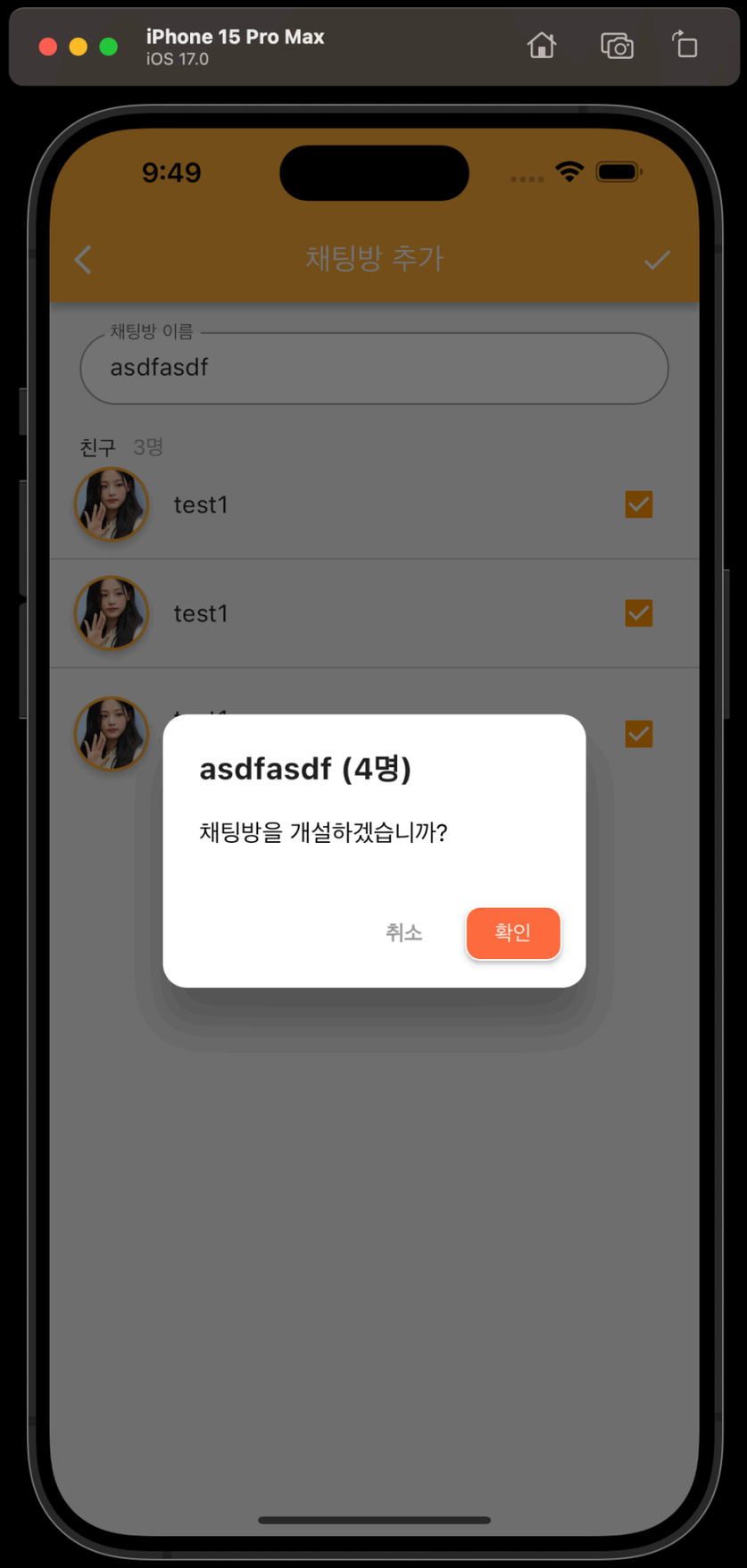
Chat Events
socketIO

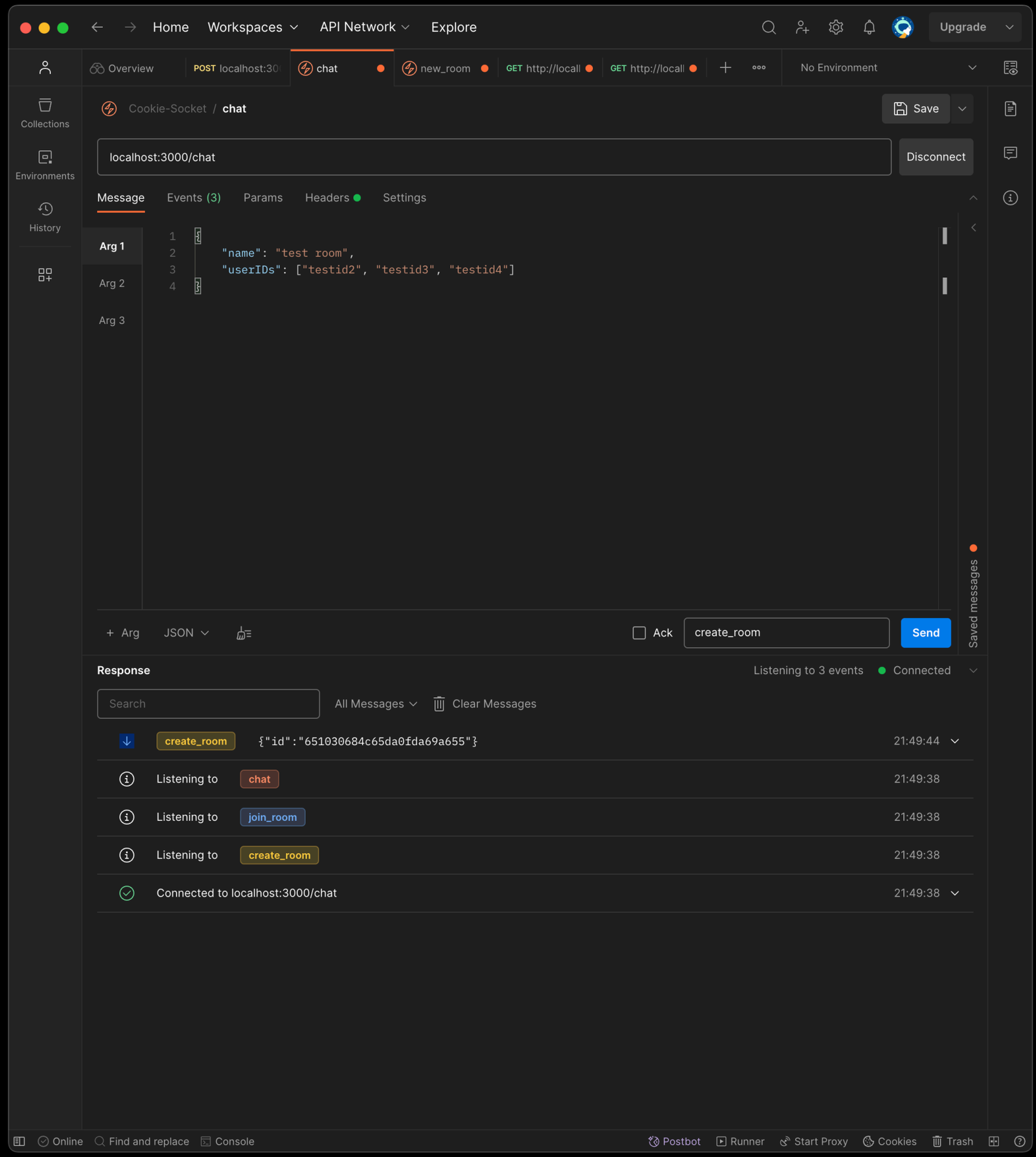
- `create_room`
- `invite_room`
- `join_room`
- `leave_room`
- `chat`

Creating Room

`create_room` process




iPhone 15 Pro Max
iOS 17.0
9:49
채팅방 추가
채팅방 이름
asdfasdf
친구 3명
test1
test1
asdfasdf (4명)
채팅방을 개설하겠습니까?
취소
확인


Home
Workspaces
API Network
Explore
Upgrade
Overview
POST localhost:30
chat
new_room
GET http://locall
GET http://locall
No Environment
Collections
Environments
History
Cookie-Socket / chat
Save
localhost:3000/chat
Disconnect
Message
Events (3)
Params
Headers
Settings
Arg 1
Arg 2
Arg 3
{
"name": "test room",
"userIDs": ["testid2", "testid3", "testid4"]
}
+ Arg
JSON
Ack
create_room
Send
Saved messages
Response
Listening to 3 events
Connected
Search
All Messages
Clear Messages
create_room
{"id":"651030684c65da0fda69a655"}
21:49:44
Listening to
chat
21:49:38
Listening to
join_room
21:49:38
Listening to
create_room
21:49:38
Connected to localhost:3000/chat
21:49:38
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash

Typed Mongoose

Infer Type from Mongoose Schema

```
30
 1 export const AccountSchema = new Schema(
 2   {
 3     _id: {
 4       type: String,
 5     },
 6     password: {
 7       type: String,
 8       required: true,
 9     },
10     name: {
11       type: String,
12       required: true,
13       unique: false,
14     },
15     birthday: {
16       type: Date,
17       required: true,
18     },
19     phone: {
20       type: String,
21       required: true,
22       unique: true,
23     },
24     profile: {
25       type: ProfileSchema,
26       required: true,
27     },
28     friendIDs: [
29       {
30         type: String,
31         ref: "Account",
```

```
 TS index.ts ×   TS handler.ts ×   TS account.model.ts ×   TS chat.model.ts ×
23   ];
22 };
21
20 // methods
19 interface IAccountDocument {
18   getFriends: typeof accountSchema.methods.getFriends;
17   generateJWT: typeof accountSchema.methods.generateJWT;
16   verifyPassword: typeof accountSchema.methods.verifyPassword;
15   getProfile: typeof accountSchema.methods.getProfile;
14 }
13
12 // statics
11 interface IAccountModel extends Model<IAccountDocument> {
10   getFriends: typeof accountSchema.statics.getFriends;
 9   createAccount: typeof accountSchema.statics.createAccount;
 8   findUser: typeof accountSchema.statics.findUser;
 7 }
 6
 5 const Account: IAccountModel = mongoose.model<IAccountDocument, IAccountModel>(
 4   "Account",
 3   accountSchema
 2 );
 1
200 export default Account;
~
~
~
~
~
~
~
~
~
~
~
~
NORMAL  4e9eca3!  src/schemas/account.model.ts   typescript  utf-8[unix]  100%  ln :200/200≡%:23  E:1(L4)
parkjb 19%  16 GB  57 kB↓  14 kB↑
```

Before

account.model.ts ×

```
// Use transform to rename _id to userid (if needed)
const transform = function (doc: any, ret: any, options: any) {     ■ 'options' is declared but its value is never read.
  // ret.userid = doc._id;
  ret.id = doc._id;
  delete ret._id;
};

AccountSchema.set("toObject", { transform });
AccountSchema.set("toJSON", { transform });

export type IAccount = InferSchemaType<typeof AccountSchema>;
export type IP                                        eof ProfileSchema>;
export default                                        ccountSchema);
```

```
type IAccount = {
    name: string;
    password: string;
    birthday: Date;
    phone: string;
    profile: {
        image: string;
        message: string;
    };
    friendIDs: string[];
    chatRoomIDs: string[];
    _id?: string;
}
```

After

NORMAL   main!   src/schemas/account.model.ts   typescript   utf-8[unix]   98%   ln :143/145≡%:15

parkjb   39%   17 GB   1.0 kB↓   0.0 kB↑

Static & Instance Methods Type Inference

Before (JS style)

```
  return jwt.sign(
    {
      userid: this._id,
      username: this.username,
    },
    process.env.JWT_SECRET_KEY!,
    {
      algorithm: "HS256",
      expiresIn: "7d",
      issuer: process.env.BASE_URI,
      subject: "userInfo",
    }
  );
};

accountSchema.methods.verifyPassword = function (password: string) {
  const hashed = hash(password);
  return this.password === hashed;
};

accountSchema.methods.getProfile = function () {
  return this.profile;
};

accountSchema.statics.getFriends = async function (userid) {
  const result = await Account.aggregate([
    {
      $match: {
        _id: userid,
      },
    },
```

NORMAL 4e9eca3! src/schemas/account.model.ts typescript utf-8[unix] 73% ln :146/200≡%:6

```
10      if (verified)
9         return res.status(200).json({
8           success: verified,
7           account: {
6             userid: account._id,
5             username: account.username,
4             phone: account.phone,
3             friendList: await account.getFriends(),
2             profile: account.profile,
1           },
40          access_token: account.generateJWT(),
1         });                              any
2       else
3         return res
4           .status(401)
5           .json({ success: verified, message: "Unauthorized" });
6     } catch (e) {
7       console.error(e);
8       return res.status(500).send({ message: "Internal Server Error" });
9     }
10  });
11
12  export default router;
~
~
~
~
~
~
~
~
~
~
NORMAL  4e9eca3!  src/routes/auth/signin.ts                typescript  utf-8[unix]  76%  ln :40/52≡%:37  E:1(L25)
```

After (TS Style)

```
26      timestamps: true,
25      query: {
24        byPhone: function (phone: string) {
23          return this.findOne({ phone });
22        },
21      },
20      statics: {
19        async createAccount({
18  +---  6 lines: userid: _id,·····
17        }) {
16  +--- 14 lines: const account = new this({·····
15        },
14      },
13      methods: {
12        generateJWT() {
11  +--- 13 lines: return jwt.sign(·····
10        },
9         verifyPassword(password: string) {
8   +---  2 lines: const hashed = hash(password);·····
7         },
6         async getFriends() {
5   +---  3 lines: return (await this.populate("friendIDs", "_id name profile"))[·····
4         },
3       },
2     }
1   );
132
1   // Use transform to rename _id to userid (if needed)
H   2   const transform = function (doc: any, ret: any, options: any) {     ■ 'options' is declared but its value is never read.
3     // ret.userid = doc._id;
4     ret.id = doc._id;
NORMAL  main!  src/schemas/account.model.ts                    typescript  utf-8[unix]  91%  ln :132/145≡%:1
```

```
14    const account = await Account.findById(userid);
13    if (!account) return res.status(401).json({ message: "User not found" });
12
11    const verified = account.verifyPassword(password);
10    if (verified)
 9      return res.status(200).json({
 8        success: verified,
 7        account: {
 6          id: account._id,
 5          name: account.name,
 4          phone: account.phone,
 3          profile: account.profile,
 2          friendList: await account.getFriends(),
 1        },
40        access_token: account.generateJWT(),
 1      });
 2    else
 3      return res
 4        .status(401)
 5        .json({ success: verif
 6  } catch (e) {
 7    console.error(e);
 8    return res.status(500).sen
 9  }
10 });
11
12 export default router;
~
~
~
~
```

```
(method) generateJWT(this: Document<unknown, {}, {
    name: string;
    password: string;
    birthday: Date;
    phone: string;
    profile: {
        image: string;
        message: string;
    };
    friendIDs: string[];
    chatRoomIDs: string[];
    _id?: string;
}> & {
    name: string;
    password: string;
    birthday: Date;
    phone: string;
```

Importance of Official API Documents

Thank You

Questions are welcome