

Ingeniería de software 2

Workshop No. 3 - Full Stack Implementation and Testing



Profesor:

Carlos Andres Sierra Virguez

Integrantes:

Alejandro Argüello Muñoz
Juan Sebastián Umaña Camacho
Juan Luis Vergara Novoa
Tomás Felipe Garzón Gómez
Tomás Sebastián Vallejo Fonseca
Juan Diego Velasquez Pinzon

Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industrial
Universidad Nacional de Colombia

Bogotá D.C

1. Database implementation

The application integrates three different databases, each serving a distinct purpose within the platform's architecture: Clerk, Stream, and a custom PostgreSQL database managed with Prisma ORM. All external services (Clerk and Stream) are accessed exclusively via REST APIs, while the PostgreSQL database is accessed through both REST endpoints and the Prisma Client ORM.

Together, these databases support a scalable video conferencing and social networking platform with features such as real-time communication, friend management, role-based access control (RBAC), and strict audit logging.

1.1. Clerk Database

Clerk is responsible for authentication and identity management. We rely on Clerk's Users table as the single source of truth for all user identities. Each user is identified by a Clerk User ID, which is referenced by Stream and the PostgreSQL database to maintain consistent cross-service identity mapping.

1.2. Stream Database

The stream tables that our application uses are:

- Call – Represents live or historical video calls.
- Recordings – Stores recordings associated with call sessions.
- Users – A representation of platform users within Stream, directly linked to Clerk's Users table by using the Clerk User ID as the primary key.

All interactions with stream tables are performed through the Stream REST API, enabling low-latency communication features and event-driven updates.

1.3. PostgreSQL Database (Prisma ORM)

The core application data model resides in a PostgreSQL database managed through Prisma ORM. The schema follows Third Normal Form (3NF), ensuring a clean, scalable, and maintainable structure suitable for a distributed production environment. All primary keys are UUIDs to enhance security and support horizontal scaling. This database stores project-specific entities, including the friend connection system, RBAC data, and auditable records of user actions.

1.3.1. Bidirectional & Social Relationships

The system's relational design allows for self-referential Many-to-Many relationships on the User entity. This structure is used to model social connections (friends, acquaintances, blocked users) in a scalable, database-normalized way.

1.3.2. Connection State Machine

A strict state machine governs the lifecycle of user-to-user relationships:

- PENDING – A request has been sent.
- ACCEPTED – Users are connected.
- BLOCKED – One user has restricted contact.

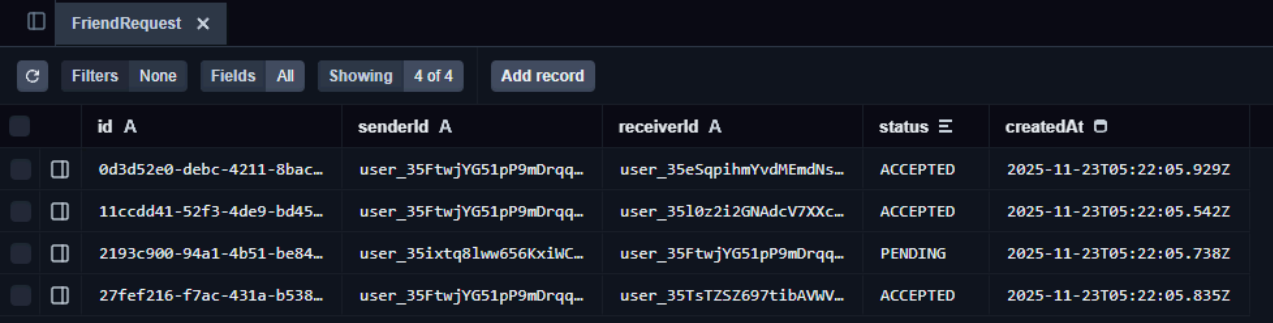
The state machine is enforced by:

- Enums restricting invalid states.
- Composite unique constraints preventing duplicate or conflicting relationships.

This guarantees consistency and predictability throughout friendship and connection workflows.

1.4. Sample data for testing

- Example table: FriendRequest
- Seeding command: `npx prisma db seed`
- Database inspection command: `npx prisma studio`



The screenshot shows the Prisma Studio interface for the `FriendRequest` table. The table has six columns: `id`, `senderId`, `receiverId`, `status`, and `createdAt`. There are four rows of data, each with a checkbox in the first column. The data is as follows:

	id A	senderId A	receiverId A	status	createdAt
<input type="checkbox"/>	0d3d52e0-debc-4211-8bac...	user_35FtwjYG51pP9mDrqq...	user_35eSqihmYvdMEmdNs...	ACCEPTED	2025-11-23T05:22:05.929Z
<input type="checkbox"/>	11ccdd41-52f3-4de9-bd45...	user_35FtwjYG51pP9mDrqq...	user_35l0z2i2GNAdcV7XXc...	ACCEPTED	2025-11-23T05:22:05.542Z
<input type="checkbox"/>	2193c900-94a1-4b51-be84...	user_35ixtq8lw656KxiWC...	user_35FtwjYG51pP9mDrqq...	PENDING	2025-11-23T05:22:05.738Z
<input type="checkbox"/>	27fef216-f7ac-431a-b538...	user_35FtwjYG51pP9mDrqq...	user_35TsTZSZ697tibAVWV...	ACCEPTED	2025-11-23T05:22:05.835Z

Figure 1. Running prisma studio with sample data seed

1.5. Database Diagram

1.5.1. Attached File Name: [DBmodel.pdf]

2. Backend services

Some examples of backend implementation:

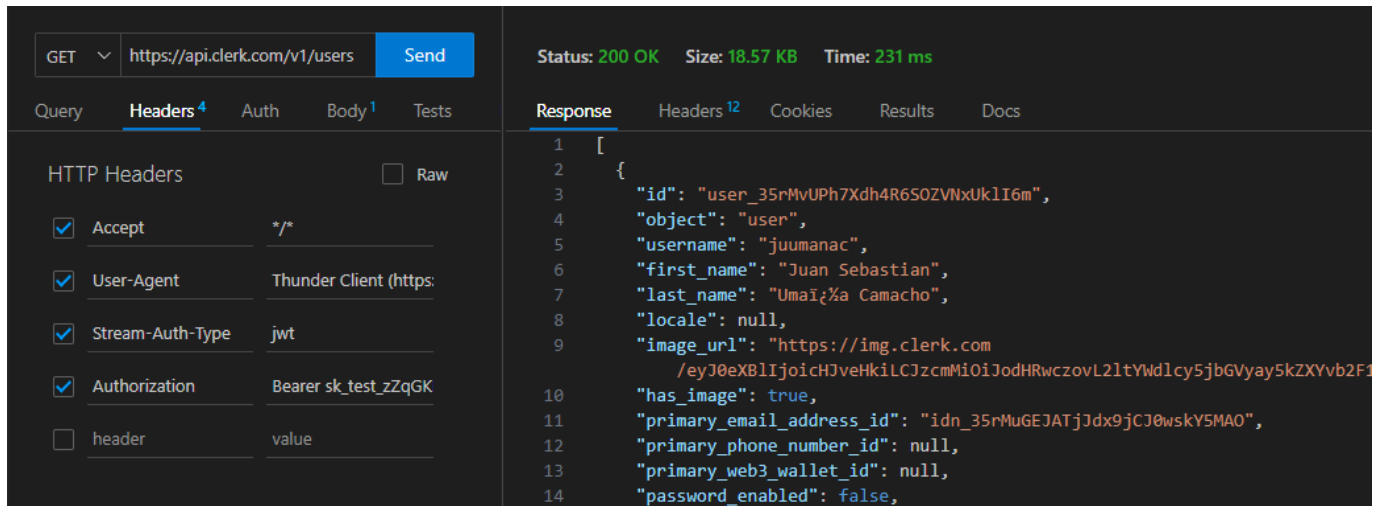


Figure 2. Get all clerk users

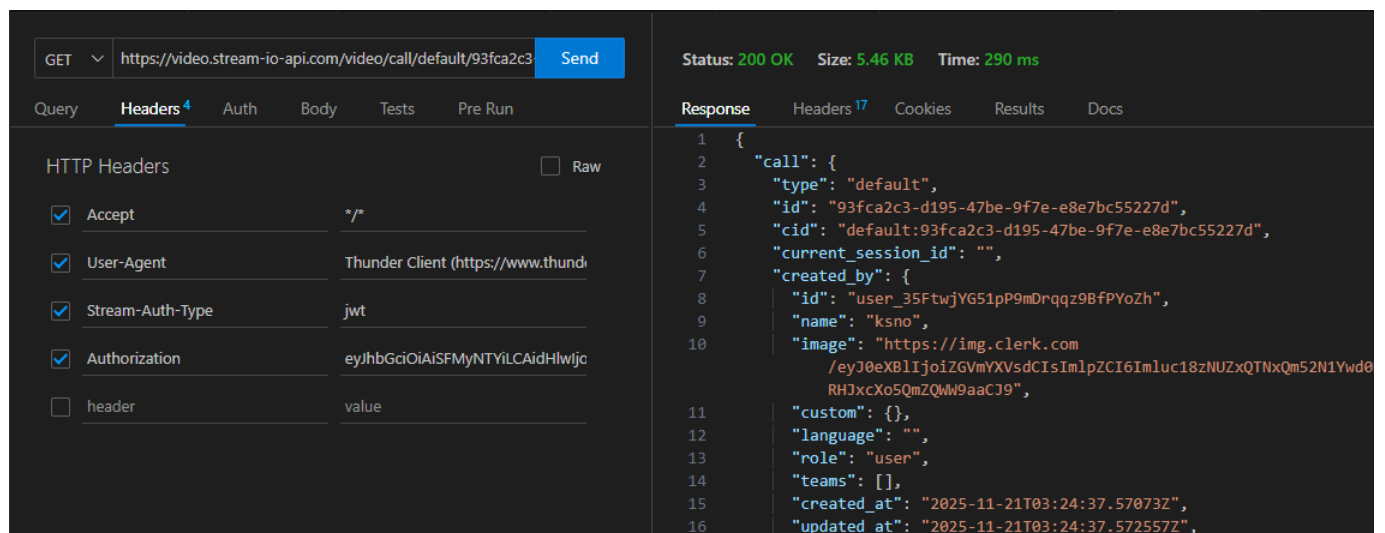


Figure 3. Get a stream call

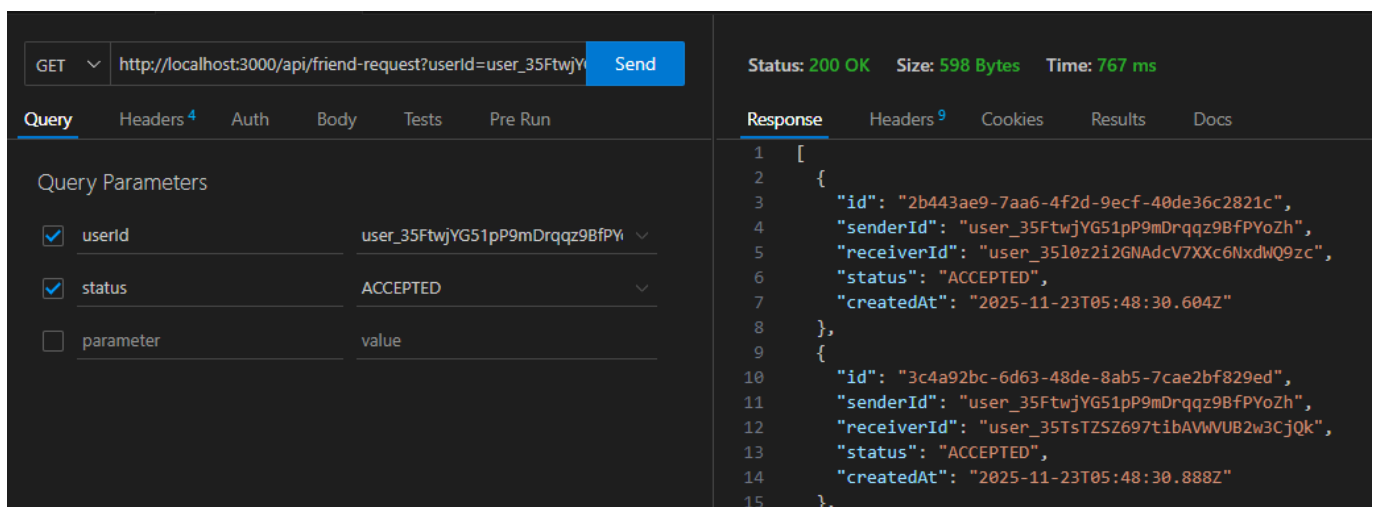


Figure 4. Get an user friends

3. *Web Frontend*

Talk2

Connect. Talk. Collaborate with **Talk2**

Talk2 makes video meetings effortless. Whether you're working with your team, meeting clients, or catching up with friends, enjoy secure and reliable video calls anytime, anywhere.

Get Started



Why choose **Talk2**?



HD Video Calls

Enjoy crystal-clear video quality for all your meetings.



Real-Time Chat

Stay connected with instant messaging during calls.



Screen Sharing

Share your screen seamlessly for presentations and reviews.

Figure 5. Landing Page

Talk2



- Dashboard
- Friends
- Upcoming
- Previous
- Recordings
- Personal Room

Upcoming Meeting: 07:48 PM

12:53 AM

Sunday, November 23, 2025



New Meeting

Start an instant meeting



Join Meeting

via invitation link



Schedule Meeting

Plan your meeting

Figure 6. Dashboard

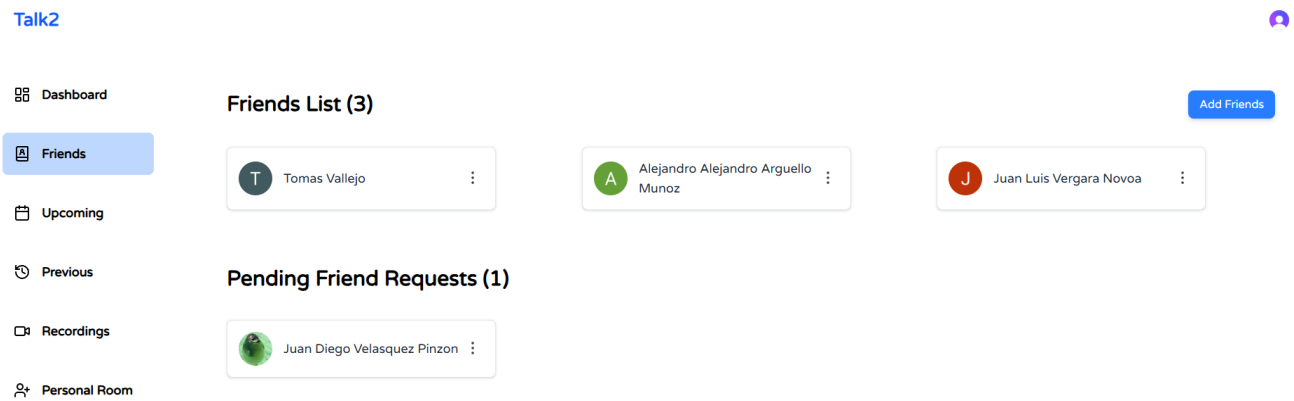


Figure 7. Friends tab

4. Unit Testing

The application includes a test suite implemented using Cypress, focusing on the complete CRUD lifecycle of the API'S. These tests validate backend correctness, input validation, state-machine transitions, and error handling. Before each test, the database is reset using `cy.resetDB()` to ensure deterministic and isolated execution.

- Testing library: Cypress
- Example test: FriendRequest Full Crud

```
PS C:\Users\ksno\Desktop\Dev\Ingesoft 2\Final-Project> npx cypress run

DevTools listening on ws://127.0.0.1:52680/devtools/browser/1266040f-38c6-462c-9db2-4c37b8f4917d
>
=====

(Run Starting)

Cypress:      15.7.0
Browser:      Electron 138 (headless)
Node Version: v22.21.1 (C:\Program Files\nodejs\node.exe)
Specs:        5 found (create.cy.ts, delete.cy.ts, full-crud.cy.ts, read.cy.ts, update.cy.ts)
Searched:    cypress/e2e/**/*.cy.{js,jsx,ts,tsx}
```

Figure 8. Unit test startup

```
Running: create.cy.ts (1 of 5)
>

FriendRequest - CREATE
  ✓ should create a new friend request (4109ms)
  ✓ should return existing request if it already exists (1585ms)
  ✓ should return 400 if missing currentUserId (756ms)
  ✓ should return 400 when sending request to yourself (591ms)

4 passing (7s)

(Results)

Tests:      4
Passing:    4
Failing:    0
Pending:    0
Skipped:    0
Screenshots: 0
Video:      false
Duration:   7 seconds
Spec Ran:   create.cy.ts
```

```
Running: delete.cy.ts (2 of 5)

FriendRequest - DELETE
  ✓ should delete an existing request (1516ms)
  ✓ should return 404 if no request exists (791ms)

2 passing (2s)

(Results)

Tests:      2
Passing:    2
Failing:    0
Pending:    0
Skipped:    0
Screenshots: 0
Video:      false
Duration:   2 seconds
Spec Ran:   delete.cy.ts
```

Figure 9. Create and delete unit test

Running: full-crud.cy.ts

(3 of 5)

FriendRequest - FULL CRUD FLOW

✓ should perform full CRUD flow (4355ms)

1 passing (4s)

(Results)

Tests:	1
Passing:	1
Failing:	0
Pending:	0
Skipped:	0
Screenshots:	0
Video:	false
Duration:	4 seconds
Spec Ran:	full-crud.cy.ts

Running: read.cy.ts

(4 of 5)

FriendRequest - LIST

✓ should list pending friend requests for a user (1464ms)

✓ should return 400 if missing query params (641ms)

2 passing (2s)

(Results)

Tests:	2
Passing:	2
Failing:	0
Pending:	0
Skipped:	0
Screenshots:	0
Video:	false
Duration:	2 seconds
Spec Ran:	read.cy.ts

Figure 10. Full crud flow and get (list) unit test

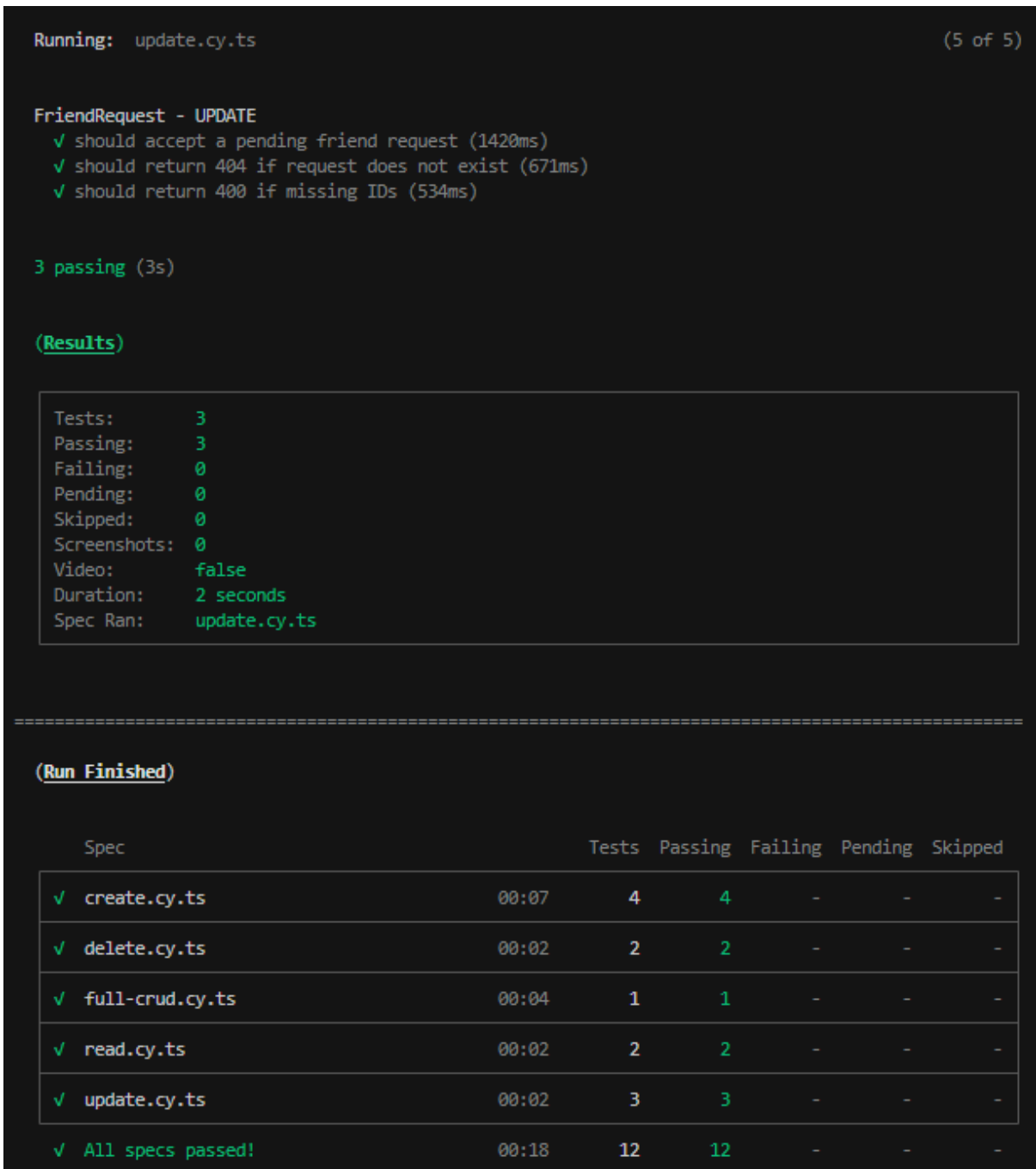


Figure 11. Update test and unit test summary

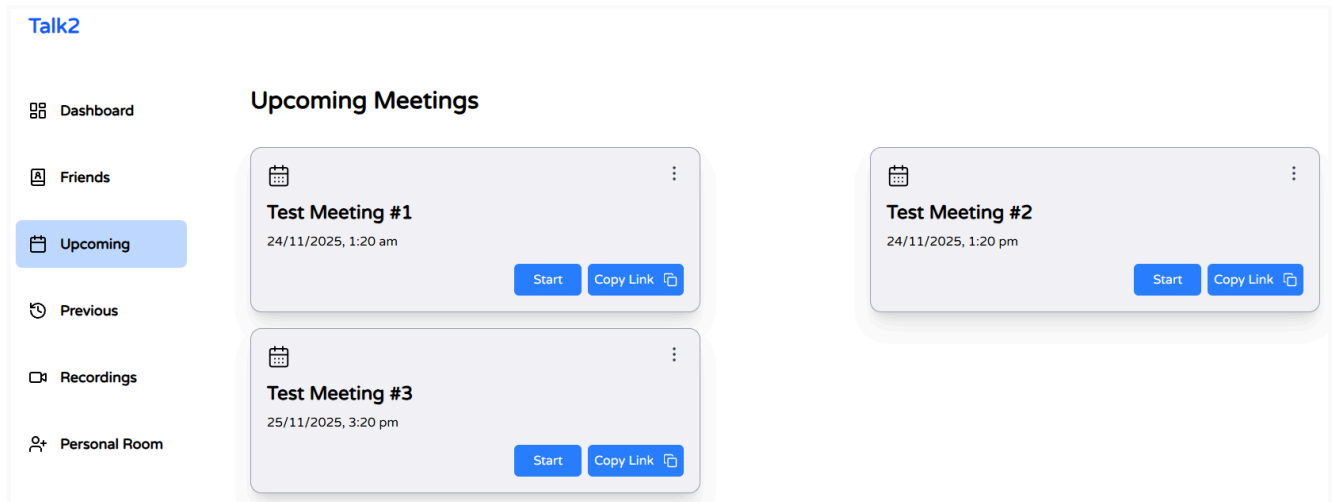
5. *Integration Evidence*

Some examples of backend-frontend integration in the app:

Backend Endpoint (Get call):

- GET `https://video.stream-io-api.com/video/call/{type}/{id}`

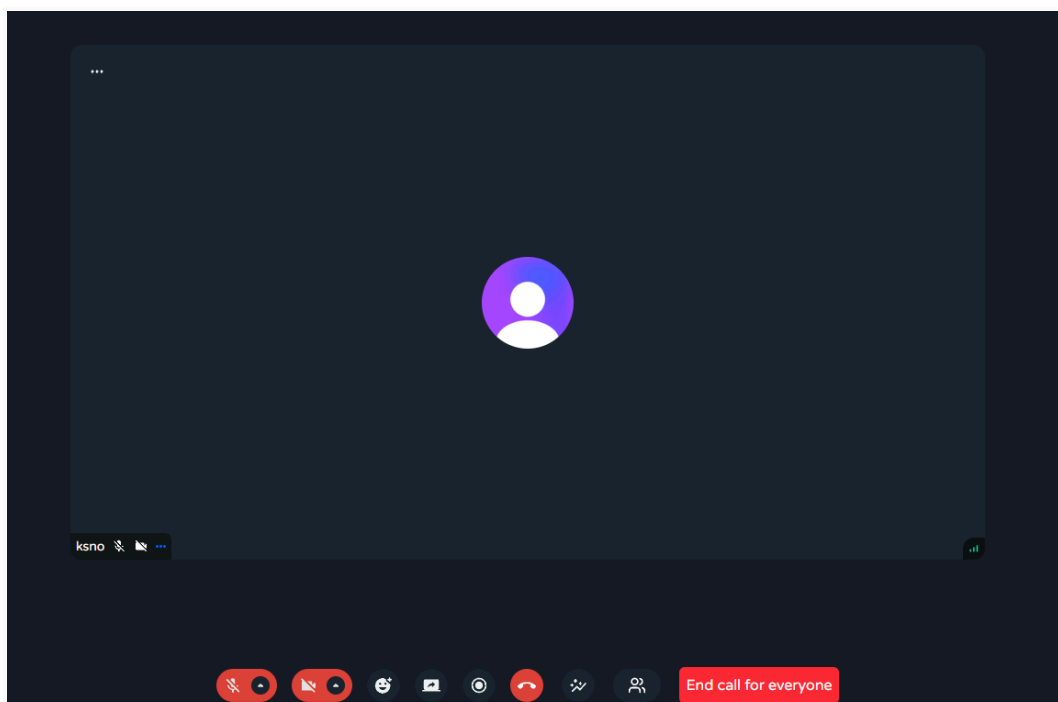
Frontend (Upcoming tab):



Backend Endpoint (Create call):

- POST `https://video.stream-io-api.com/video/call/{type}/{id}`

Frontend (Call Room):



Backend Endpoint (GET users):

- GET <https://api.clerk.com/v1/users>

Frontend (Add friends modal):

