Real-time multiple user drawing tools demo

Doan Quang Tuyen

Demo url and demo code

1. Demo url: https://www.loom.com/share/6a62c03df6894a11be21850fd082f9d8?sid=93e2724f-d1b8-4948-a4e3-7a8b3ad6acdb
2. Frontend code: https://github.com/doannucphys/drawing_frontend/tree/main
3. Backend code: https://github.com/doannucphys/drawing_websocket
4. Document: https://github.com/doannucphys/drawing_docs

Outline

1. Requirements & assumption
2. Architecture Diagram
3. Component Description
4. Data Flow
5. Technologies
6. Implementation and demo
7. Discussion for future & extension

1 - Requirements & assumption

Requirements

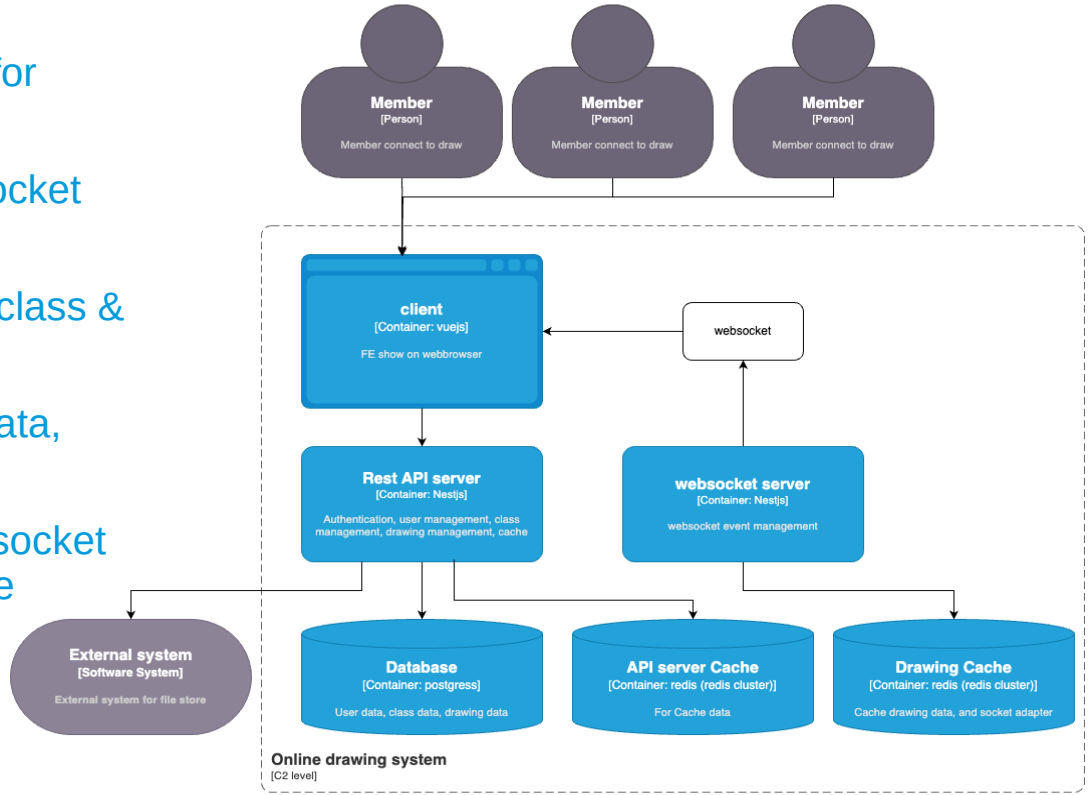
- Real-time drawing system for multiple user
- User register, login, logout, forgot/reset password
- Authentication user can create class and other can connect to the same canvas to draw
- Limit number user by class is 30
- Have leaderboard show number of connected user and this list updated when user connect/leave drawing session

Assumption:

- User connect to canvas via classId
- Have leaderboard show list of connected user to drawing session
- Demo implement only Frontend code and websocket code, other parts are not implemented
- Demo not implemented authentication (login/logout/forgot password) and only demo the most challenge part – the realtime drawing

2 - Architecture Diagram

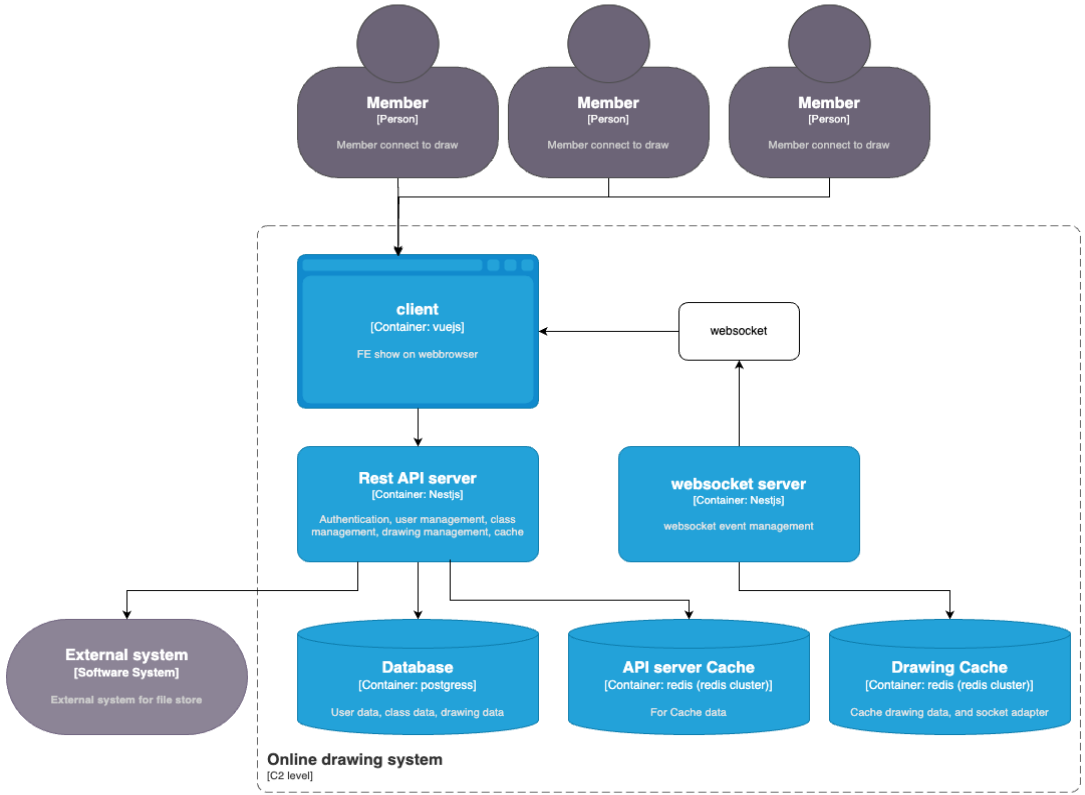
- Client (vuejs code): – Frontend of system
- Rest API server (nestjs code): provide api for authentication, quiz and user management
- Websocket server (simple nodejs code): socket event management
- Database: postgress database store user, class & drawing data
- Cache (redis): Cache user data, drawing data, authentication data
- On local, for this demo, run client and websocket server on local. For real application, we use kubernetes when deploy services within monitoring (eg. ganfana) for error log.



3 - Component Description

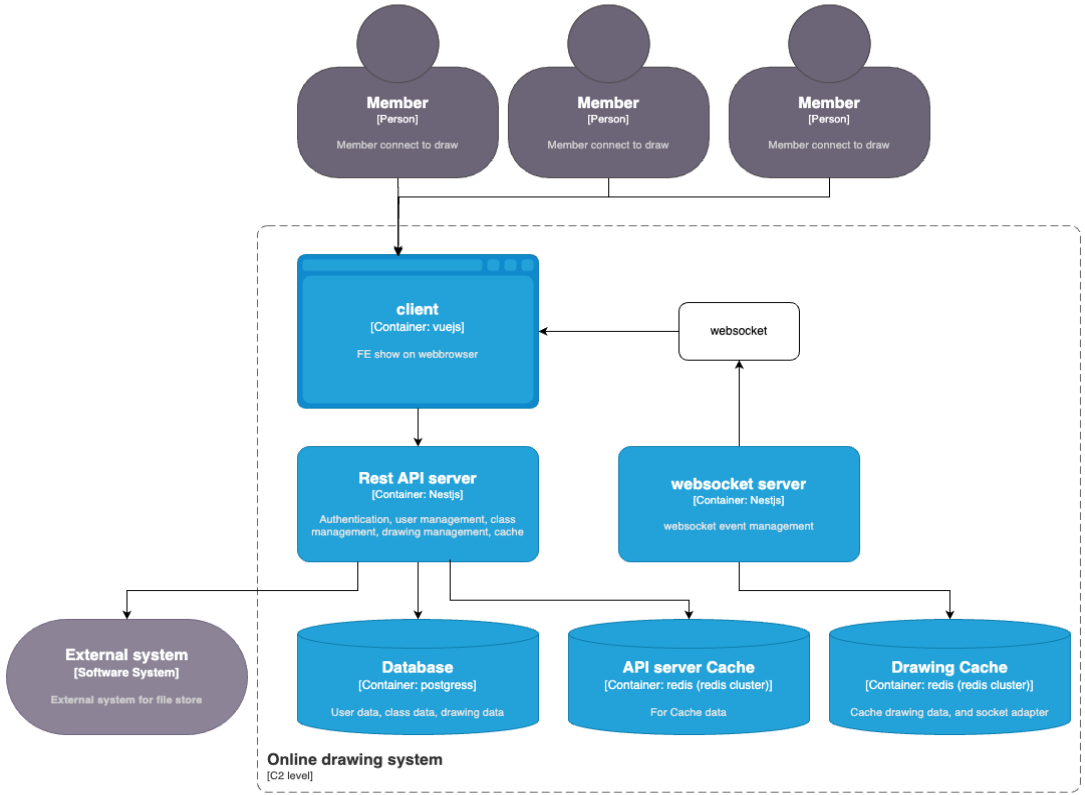
Client (vuejs code):

- Use vuejs – simple and easy to maintain, good performance
- Use tailwindcss – popular css framework, make code easy to maintain
- Implement socket for realtime update draw for all user and all user can draw on same canvans. Drawing data is manage in store and localstorage
- Authentication (Login/logout/forgot password page .. (not show in demo). For simplified demo, authentication & authorization are not implemented



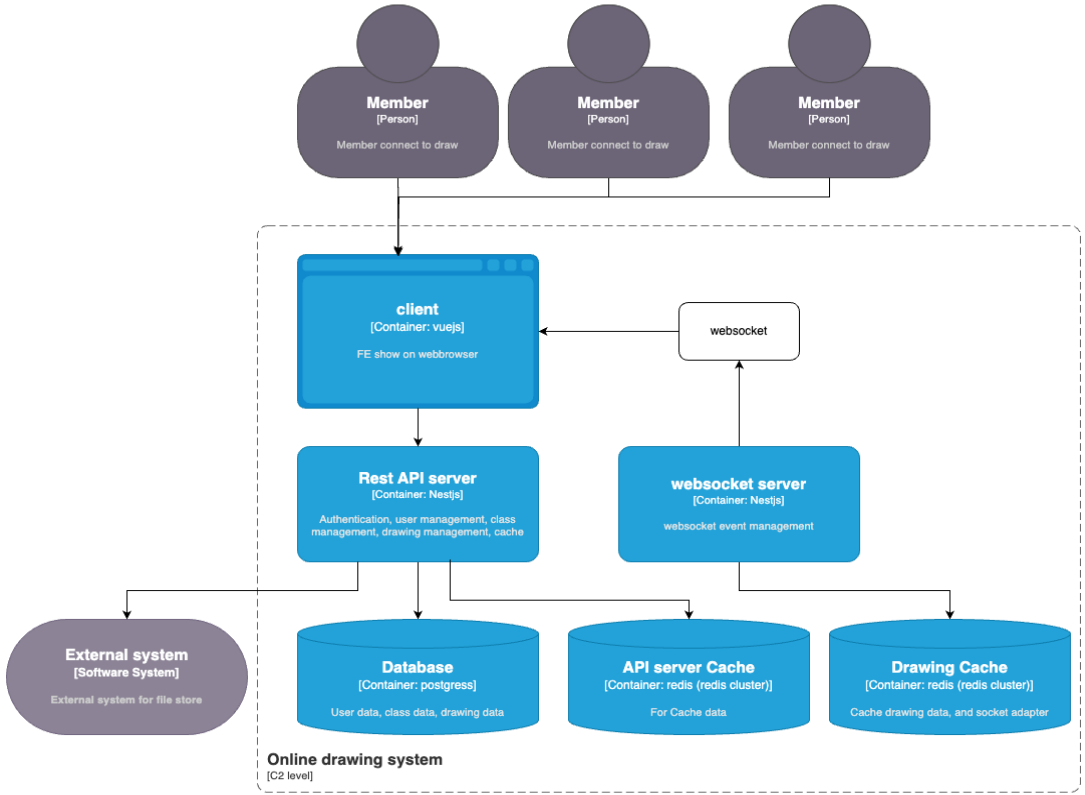
3 - Component Description

- **Websocket server (simple nodejs code): socket event management**
- For this demo, use simple nodejs code use socket.io for websocket event management
- For real application, we can use also node-thread to optimize performance and redis cluster as adapter for socket, an other technique for performance optimization.



3 - Component Description

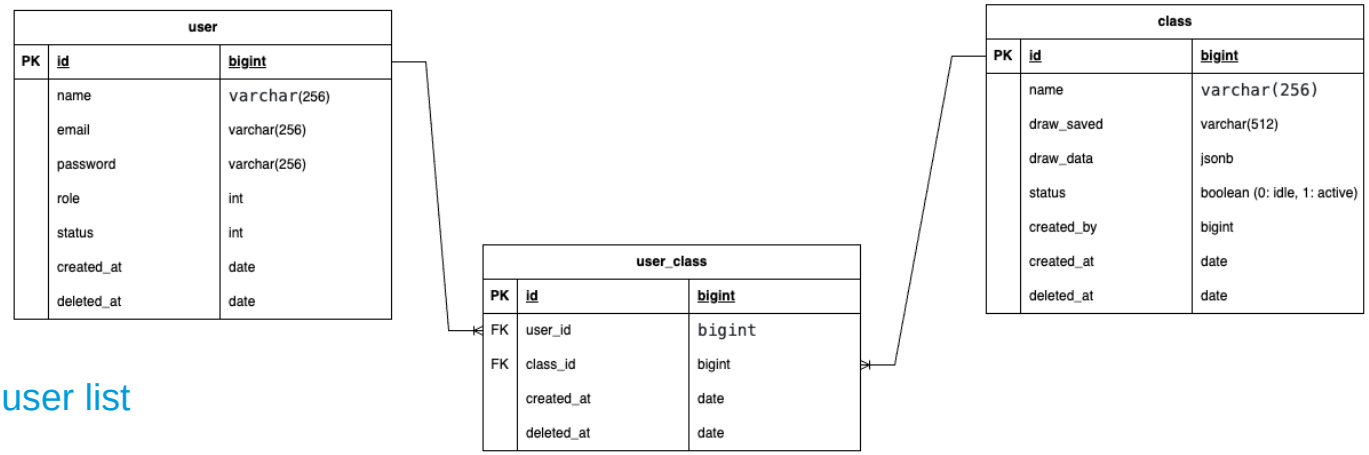
- **Rest API server (nestjs code): provide api for authentication, quiz and user management**
- For this demo, this server is not implemented
- For real application, this service is implemented with swagger, cache, authentication, authorization, user management and quiz management module. This server can contain more than 1 services depend on number user to optimize read/write performance etc. All file and media file will be store on external system (eg. Aws s3, or FTP server, cdn, ..)



3 - Component Description

Database: postgress database store user, quiz, score data, etc.

- For application can use cloud database service or onpremise database (depend on project's requirements)
- Example of db design =>
- The demo will use mock data

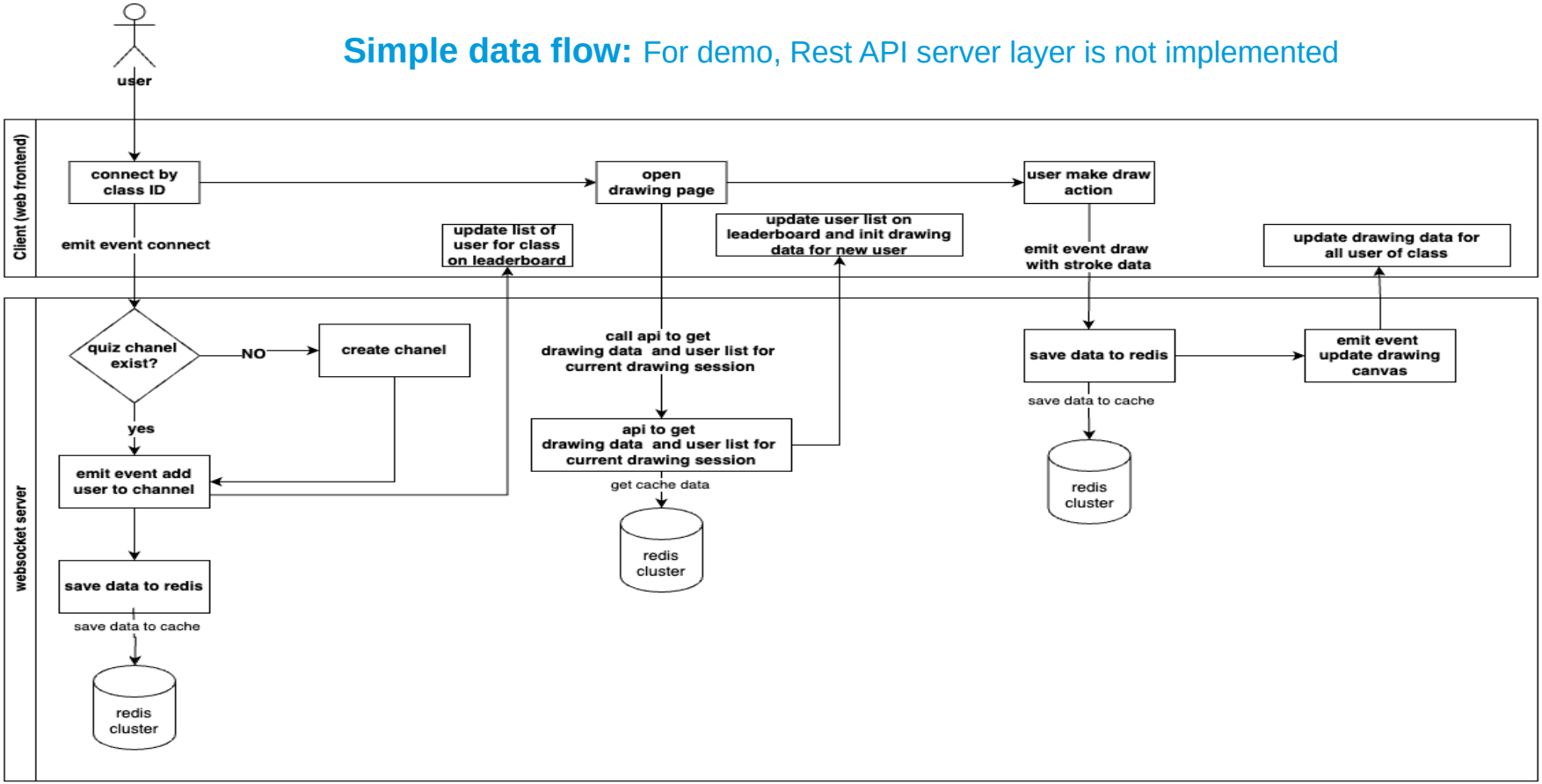


Cache (redis):

- Cache user data, authentication, authorization data, drawing data, user list for each class, socker info
- Redis cluster is also used as adapter for socket to optimize performance

4 - Data Flow

Simple data flow: For demo, Rest API server layer is not implemented



5 - Technologies

Front-end: user vuejs (vue3) with a lot of feature help development is more simple and quick, code is more clean and easy to maintain an fixbug, fully typescript support. Use tailwindcss for style css

API-service: Use Nodejs. This kind of system, have huge number of light weight request so Nodejs with the non-blocking I/O operations based on event loop mechanism is good candidate.

Use Nestjs framework - fully typescript support, code easy to maintain, fix bug and develop, auto generate swagger docs, etc

Redis for caching: Redis is good candidate, *“Redis is the most popular distributed caching engine, offering true statelessness for an application's processes, minimizing duplication of cached data, and scaling back requests to external data sources.”*

DB Postgres: a popular database with high data consistency and integrity, stable, scalability

6 - Implementation and demo

Scope:

- Demo only main function (user join class, go to drawing page, view current drawing data of current drawing session, draw, view user list and drawing update realtime)
- Not handle unit test, and other case than main functionalities
- Use mock data for jwt token is used for socket authentication check
- Implement only Client code (vuejs) and websocket server code and run one local
- New user can connect and view drawing data of current session
- Only user connect to same class view updated drawing data of that class

7 - Discussion for future & extension

- Consideration on database read/write process for optimization, database Sharding, etc.
- User node-thread and Redis for socket adapter for performance
- Optimize UX and data flow design to optimize performance
- Can develop in multi-tenant mode for multiple client width diff need/requirements
- Use message queue for some functionality, ex. sync data for read/write dbs