Real-Time Vocabulary Quiz Coding Challenge

Doan Quang Tuyen

For Elsa coding challenges requirements:
https://github.com/elsa/coding-challenges.

Demo url and demo code

1. Demo url: https://www.loom.com/share/81e70f894c1f47c9b04dca64a6e07689?sid=6f205679-1c77-47f6-aa41-2c4b48f8dd2e
2. Frontend code: https://github.com/doannucphys/quiz_frontend/tree/main
3. Backend code: https://github.com/doannucphys/quiz_websocket
4. Document: https://github.com/doannucphys/quiz_docs

Outline

1 - Requirements & assumption

Requirements

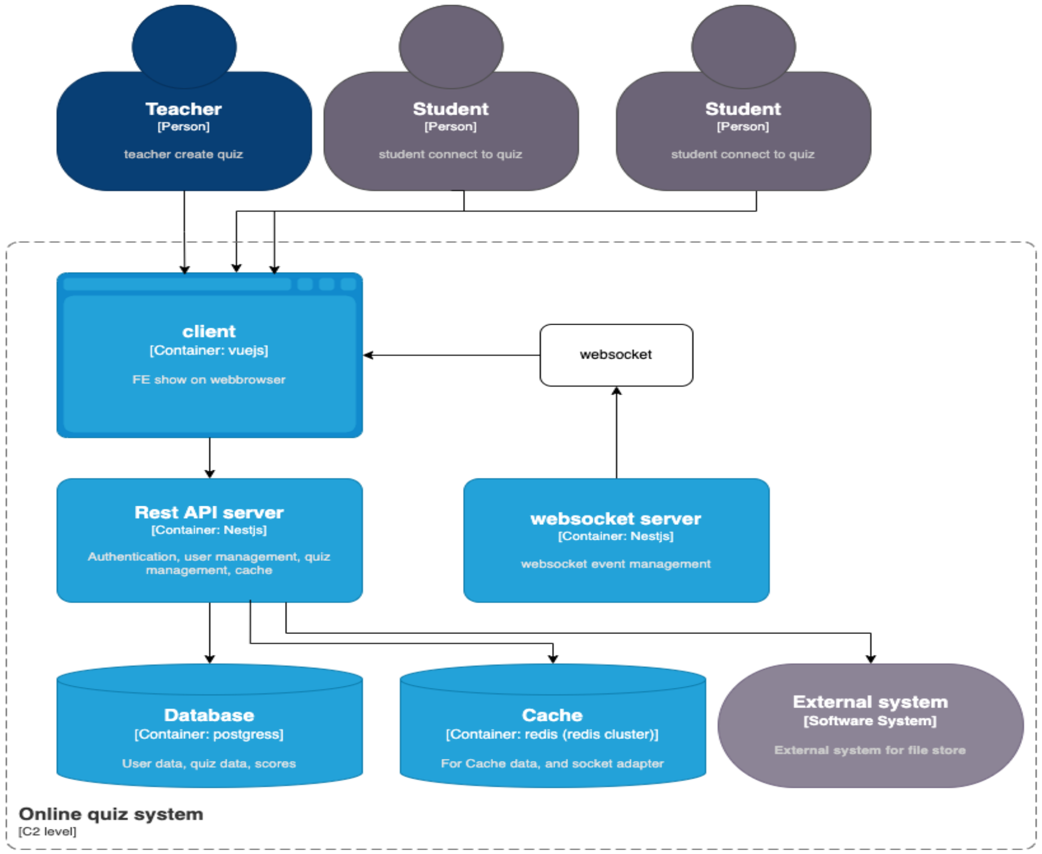
- Real-time quiz participation: User join quiz by unique quiz ID
- Real-time Score Updates: Users’ scores should be updated in realtime as they submit answers
- Real-time Leaderboard: A leaderboard should display the current standings of all participants in real-time

Assumption:

- Data update to database only when user finish all question and submit result
- Leaderboard is shown on screen of every user join the quiz. User see the scores of other users who join to the same quiz
- Demo only run for simple flow: user join quiz – do quiz – submit result and view leaderboard. All authentication process and quiz management is not include in to demo
- Demo implement only Frontend code and websocket code, other parts are not implemented and mock data will be used

2 - Architecture Diagram

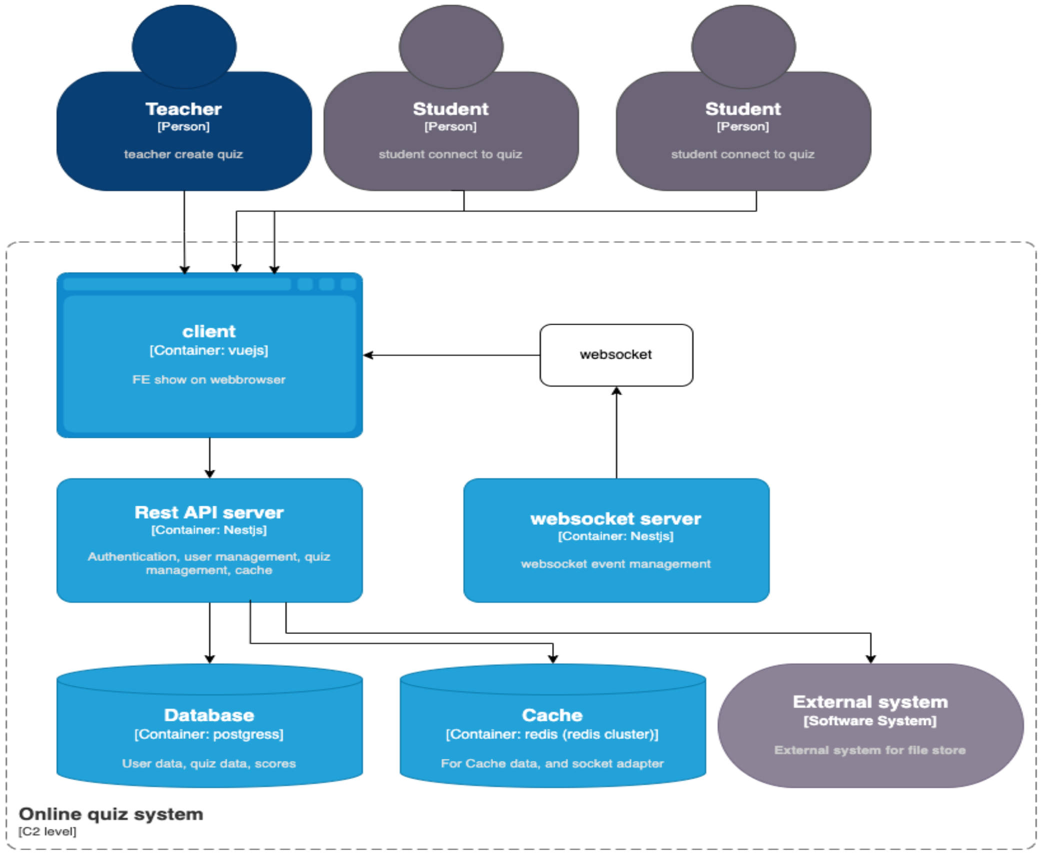
- Client (vuejs code): – Frontend of system
- Rest API server (nestjs code): provide api for authentication, quiz and user management
- Websocket server (simple nodejs code): socket event management
- Database: postgress database store user, quiz, score data
- Cache (redis): Cache user data, quiz data, question data for each quiz, user score list
- 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

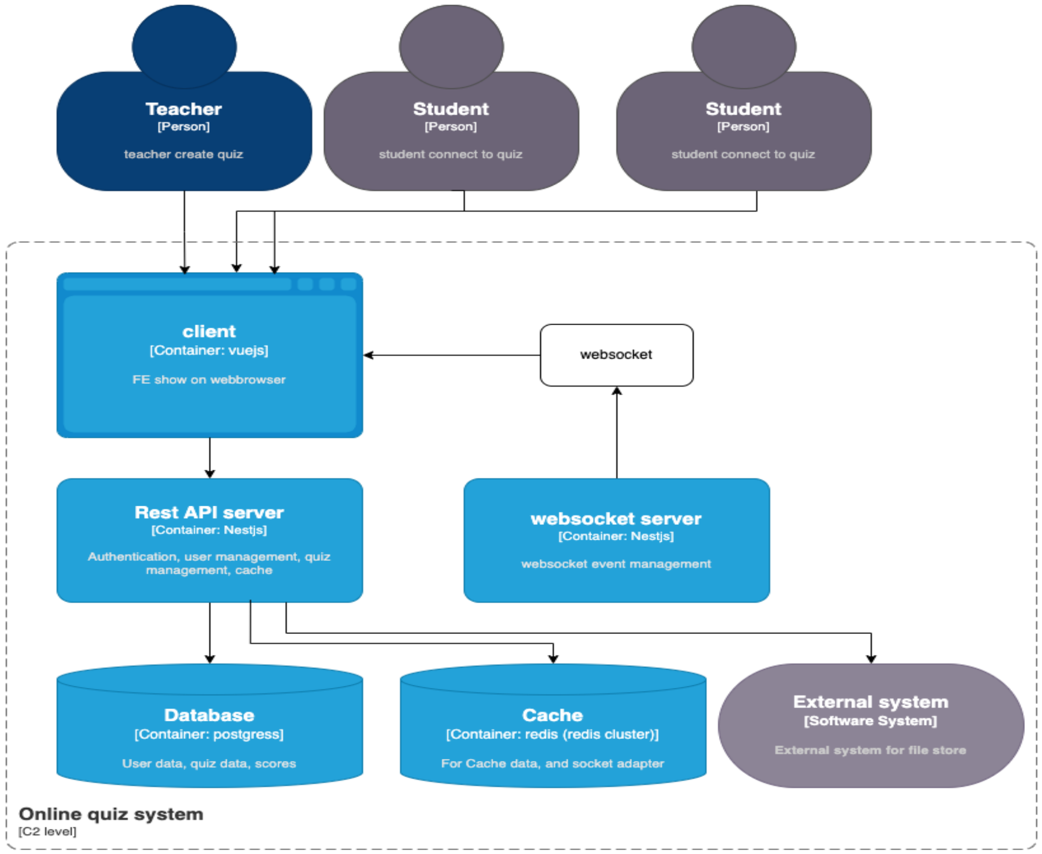
Client (vuejs code):

- Login page: (not show in demo). For simplified demo, unauthorized user can connect directly to quiz by input name, and quiz id
- Page for authenticated teacher user for quiz and question list management (not show in demo)
- Page for authenticated student user to view list quiz and connect and do quiz and leaderboard (for demo user not need to login and can go directly to this page after connect to quiz from home page)
- Implement socket for realtime update quiz score and user list for quiz
- Use tailwindcss



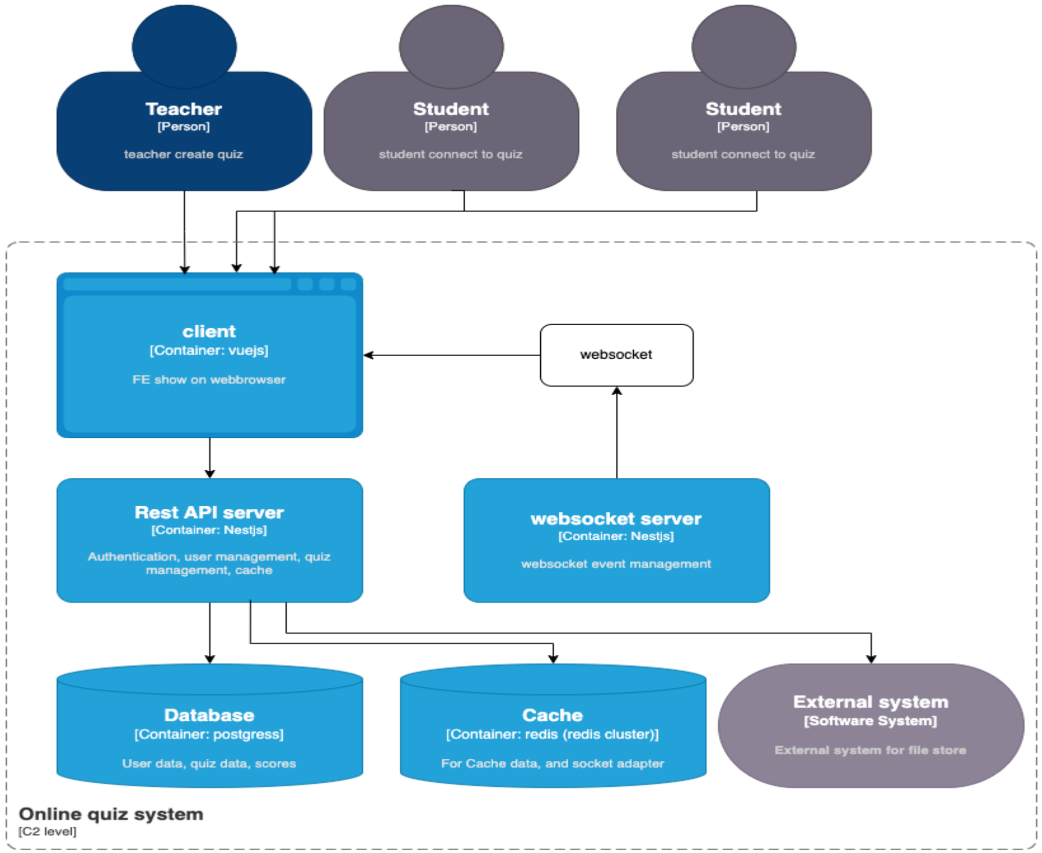
3 - Component Description

- **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

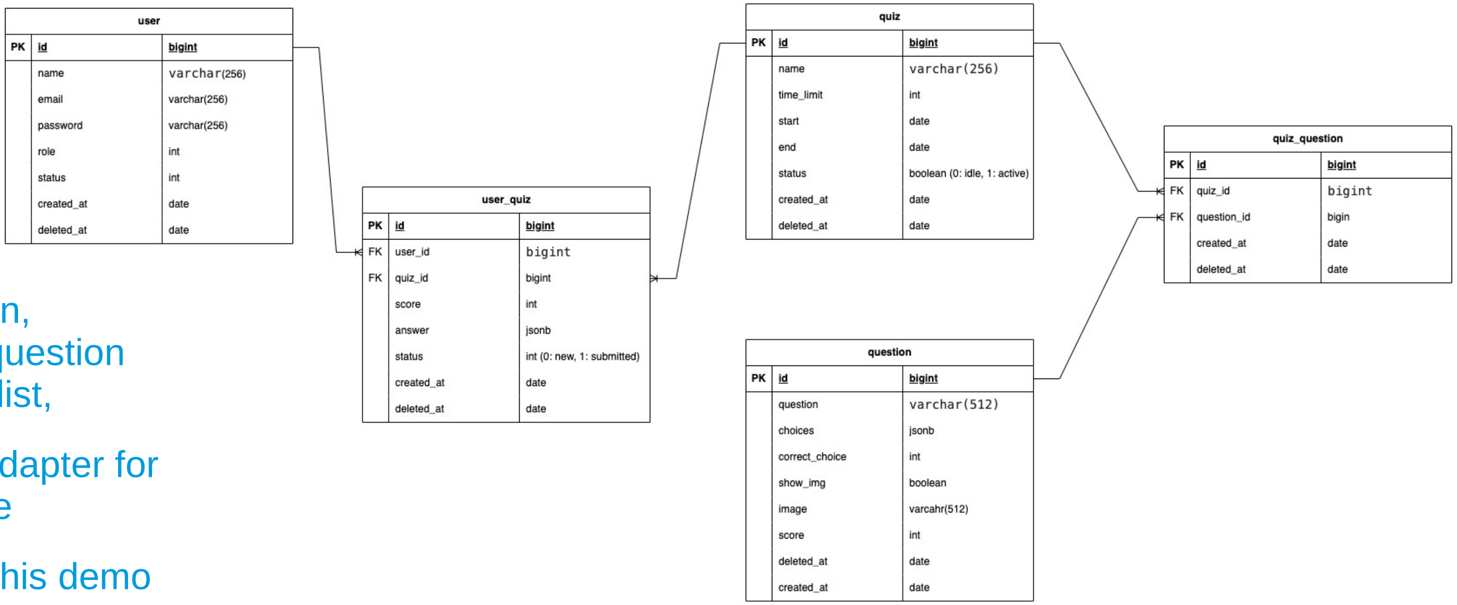
- **Rest API server (nestjs code): provide api for authentication, quiz and user management**
- For this demo, this server is not implemented and use mock data instead
- For real application, this service is implemented with swagger, cache, authentication, authorization, user management and quiz management 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



user		
PK	id	bigint
	name	varchar(256)
	email	varchar(256)
	password	varchar(256)
	role	int
	status	int
	created_at	date
	deleted_at	date

user_quiz		
PK	id	bigint
FK	user_id	bigint
FK	quiz_id	bigint
	score	int
	answer	jsonb
	status	int (0: new, 1: submitted)
	created_at	date
	deleted_at	date

quiz		
PK	id	bigint
	name	varchar(256)
	time_limit	int
	start	date
	end	date
	status	boolean (0: idle, 1: active)
	created_at	date
	deleted_at	date

quiz_question		
PK	id	bigint
FK	quiz_id	bigint
FK	question_id	bigin
	created_at	date
	deleted_at	date

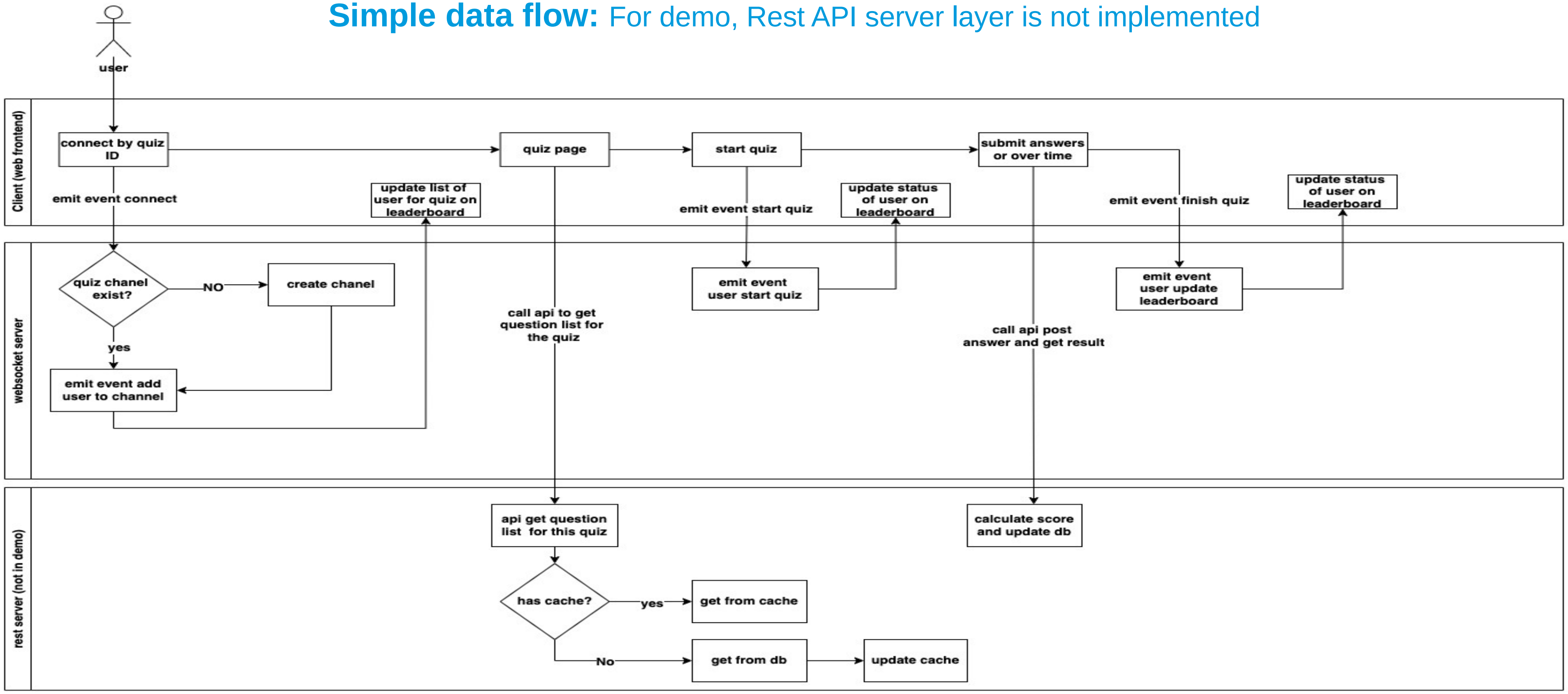
question		
PK	id	bigint
	question	varchar(512)
	choices	jsonb
	correct_choice	int
	show_img	boolean
	image	varcahr(512)
	score	int
	deleted_at	date
	created_at	date

Cache (redis):

- Cache user data, authentication, authorization data, quiz data, question data for each quiz, user score list,
- Redis cluster is also used as adapter for socket to optimize performance
- Cache is not implemented for this demo

4 - Data Flow

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



user
Client (web frontend)
connect by quiz ID
quiz page
start quiz
submit answers or over time
emit event connect
update list of user for quiz on leaderboard
emit event start quiz
update status of user on leaderboard
emit event finish quiz
update status of user on leaderboard
websocket server
quiz chanel exist?
NO
create chanel
yes
emit event add user to channel
call api to get question list for the quiz
emit event user start quiz
call api post answer and get result
emit event user update leaderboard
rest server (not in demo)
api get question list for this quiz
has cache?
yes
get from cache
No
get from db
update cache
calculate score and update db

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 quiz, do quiz, submit answer, view record list update realtime)
- Not handle unit test, and other case than main functionalities
- Use mock data for user list and question list
- Implement only Client code (vuejs) and websocket server code and run one local

Github repo for submit code:

- Frontend code:
- Websocket server code:

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 non-realtime practical test for certificate