Shaked Chen ********

OS macOS m1

Python 3.9.7

create a react app using the command npx create-react-app my-app to create the gui for the game and the server using fastapi

the server will be responsible for the game logic and the communication between the players and prisma with sql for the database

in the server we will have the option to create a question create a user create a game and join a game the game will be a kahoot game with a question and answers and the user will have to choose the right answer

the server will be responsible for the game logic and the communication between the players and prisma with sql for the database

I encrypted the user password and used industry standard for the saving the password in the database (used salt and hash) to create a secure environment for the users

I used websockets for the communication between the players and the server for most of the endpoints I added the depend on the user (for the one I don't it is only because it is much easier to test without it)

I created the clients using

```
ws_router = APIRouter()

clients: Dict[str, WebSocket] = {}
active_websockets = set()
```

there is a guide on how to use the gui

5 Theoretical Questions **

1. How can you enhance the security of your application? Discuss potential vulnerabilities and propose solutions.

– first I can encrypt the user question and not send the is_correct field to the user – I can add the depend on the user to all the endpoints after that I can add a check that the user doing the action is really the user that he says he is (take to much time to do it now)

2. What are the risks associated with transmitting messages in plain text? How might you implement encryption to address this concern?

– the risk is that the message can be read by anyone that can see the message – I can encrypt the message using a key and send the key to the user that needs to read the message – I can do it using the token that the user have, or using a

logic that the user and the server have the same key – I can use https to encrypt the message automatically

5.2 Scalability

1. How would you design your application to handle a large number of concurrent users? Discuss potential bottlenecks and scalability challenges.

– I can use a load balancer to distribute the load between the servers (for example if I have a lot of users that want to play the game) – I can use a database that can handle a large number of users (not sqlite) – I can use a cache to save the data that is used a lot (for example the questions and the answers) – I can use a message broker to handle the communication between the servers – I can use docker to create a scalable environment with auto scaling – I can upload the service to AWS, GCP etc to use their scalable environment

2. What strategies could be employed to distribute the load and balance the server's resources effectively?

– load balancer – cache – message broker – docker – auto scaling – use a scalable database – use a scalable environment like AWS, GCP etc (cloud)

5.3 Reliability and Fault Tolerance 1. Describe how you would ensure the reliability of your application. What measures can be taken to handle server failures or unexpected crashes?

– I can logs all the actions that the user do and the server do – ensure minimum amount of containers that can handle the load

2. How might you implement message persistence to ensure that messages are not lost even if the server restarts? – keep saving the data in the database – use a message broker that can save the data

5.4 User Authentication

1. Explain the importance of user authentication in your application. What methods could be used to authenticate users securely? – currently I used a token to authenticate the user that I created when the user register – I can use a 2 factor authentication – I can use a passwordless authentication (google etc)

2. How would you handle user registration and password management to enhance the overall security of the system? – I would save a cookie in the user browser to save the user token (currently I save the token in the local storage which is not secure enough)

5.5 Concurrency and Synchronization 1. Discuss the challenges associated with handling multiple simultaneous connections in a server.

– the server can crash and the data can be corrupted – hard to manage the data that is used by multiple users and not getting memory leaks when the user

disconnect and when scaling the server

2. How might you implement thread safety and synchronization to prevent data corruption or race conditions? – use well known libraries that handle the data that is used by multiple users – use message broker to handle the communication between the servers (make life easier) – use a sass that do it out of the box (like firebase or supabase)

5.6 Protocol Design

1. Explain the choice of communication protocol for your application (e.g., TCP, UDP). What factors influenced your decision? Autumn 2023

– TCP currently I used websockets to communicate between the server and the user – I choose it because it is a reliable protocol and I don't want to lose any data

2. Explain about the socket handshake – which protocol you used, what are the commands, are they blocking commands. How would you design the message format

– I used websockets to communicate between the server and the user

and protocol for communication between the client and server?

```
@ws_router.websocket("/")
async def websocket_endpoint(websocket: WebSocket):
    await websocket.accept()
    active_websockets.add(websocket)
    try:
        while True:
            data = await websocket.receive_text()

            data = json.loads(data)
            command = data.get("type")
            message = data.get("message")
            if command == "ECHO":
                response = f"Echo: {message}"


            elif command == "JOIN_TOR":
                # get the tornoment id and add the user to the tornoment
                tor_id = data.get('id')
                # add this user to the room
                if tor_id not in rooms:
                    rooms[tor_id] = set()
                rooms[tor_id].add(websocket)
```

```
            elif command == "SEND_ANSWER":
                passs


            elif command == "REVERSE":
                response = message[::-1]
            elif command == "BROADCAST":
                # Use the broadcast function to send a message to all connected clients
                # await broadcast(f"Broadcast: {message}")
                continue  # Skip adding this message to the current websocket
            else:
                response = "Unknown command"
            print(data, command, message, )

            await websocket.send_text(response)
    except WebSocketDisconnect:
        active_websockets.remove(websocket)
    finally:
        active_websockets.remove(websocket)
```

– I used the built in websocket libary in fastapi and javascript

5.7 Message Ordering and Delivery 1. How would you address the issue of message ordering and delivery in your application? Discuss potential scenarios and solutions.

– I can add a timestamp to the message and save the message in the database – I can do a order by a spesic role in the message (for example the user id)

2. What mechanisms could be employed to ensure that messages are delivered in a timely and reliable manner?

– using web sockets – using caching – using optimistic updates – using a message broker

5.8 Persistent Storage 1. Explain the importance of persistent storage in your server. How did you design and implement the storage on the server?

– I save the data in a database using prisma (sqlite) the good thing about prisma is that I can change the database to any other database without changing the code.

1. Discuss the trade-offs between different storage solutions in the context of your application.

– sqlite is good for small projects but not for big projects up to (1000 users) – I

can use postgresql for a bigger project but costly and need to manage the server advantages using postgresql: is an amazing database that can handle a lot of users and a lot of data like many big projects use it for example in my team we use postgresql for the database (fabios.io)

–I can use a nosql database if I need billions of users and a lot of data (like facebook) but it is hard to manage and need a lot of knowledge to use it also no typesaftey and join and other things that are in the sql database