Verve AI- System Design Assessment

Candidate Name: Jayraj Mulani
Candidate Email: jayrajm.mulani@gmail.com

Part 1: System Design and Integration

1. Enhancing Existing Functionalities:

1. **How would integrating a Question Bank Function enhance our current features, such as mock interviews, real-time interview copilot, and resume review?**

   Integrating a Question Bank Function into Verve AI's existing interview preparation platform can prove to be valuable to the users, especially if integrated with some of the existing features. Some of that is discussed below:

   i. Mock Interviews: By integrating the Question Bank, the system can dynamically pull questions based on various parameters like job role, difficulty level, or topic. Users can have more tailored and relevant practice sessions, with questions that closely match their target job interviews.
   ii. Interview Co-pilot: The copilot can utilize the Question Bank to offer contextually relevant questions and answers in real-time. It could also suggest follow-up questions or topics based on the user's responses.
   iii. Resume Review: The system can analyze the resume to identify key skills and experiences, then recommend specific interview questions from the Question Bank that the user should prepare for. This could lead to a more targeted interview preparation process.

2. **What are the potential benefits to users from this integration?**

   Integrating a Question Bank can potentially benefit the users in a lot of ways. Some of them are mentioned below:

   a. **Personalized Preparation**: Users would benefit from highly customized mock interviews, with questions tailored to their specific needs, leading to more effective preparation.
   b. **Comprehensive Coverage**: The Question Bank could offer a wide range of questions covering various topics and difficulty levels, ensuring users are well-prepared for different aspects of an interview.
   c. **Enhanced Learning Experience**: With the Question Bank integration, users can receive instant feedback on their answers, along with explanations, further enhancing their learning and preparation process.
   d. **Efficient Learning**: Users can focus their preparation on areas where they need the most improvement, making their study time more efficient.

3. **Suggest any new features that could leverage the Question Bank to improve user experience.**

   Some new features that may leverage the Question bank are as follows:

   a. **Adaptive Learning Paths:**
      Based on the user's performance in mock interviews and real-time interactions, the platform can create adaptive learning paths. These paths would adjust dynamically, pulling from the Question Bank to focus on the user's weak areas and progressively introduce more challenging questions.

   b. **Daily Question Challenge:**
      Users can receive daily questions based on their current preparation focus. This could be a single question or a small set of questions from the Question Bank, encouraging consistent practice.

   c. **Community-Sourced Questions:**
      Allow users to contribute to the Question Bank by submitting questions they encountered in real interviews. This would keep the Question Bank up-to-date with the latest industry trends and challenges.

   d. **Peer Review and Discussion:**
      Enable users to review and discuss questions from the Question Bank in a community forum. Users could see how others approached the same question, fostering collaborative learning.

   e. **Interview Question Trails:**
      Create a feature where users can follow a specific "trail" of questions focused on a particular topic or role. Each trail would consist of a curated set of questions from the Question Bank, gradually increasing in complexity.

2. Scalability and Performance

1. **Considering the potential increase in questions and users, what strategies would you propose to ensure scalability and performance of the Question Bank?**

   To ensure the scalability and performance of the Question Bank, especially as the number of questions and users grows, we can implement several strategies across different layers of the system.

a. **Database Optimization:**
   i. Sharding: Distribute the database across multiple shards by key (e.g., category) to balance the load.
   ii. Indexing: Implement efficient indexes on frequently queried fields to speed up searches.
   iii. Partitioning: Break down large tables into smaller segments for faster access.

iv. Query Optimization: Regularly refine database queries to maintain speed and efficiency.

b. Caching:

i. In-Memory Caching: Use Redis or “Memcached” for frequently accessed data to reduce database load.
ii. CDN: Cache static assets (e.g., images) closer to users for faster content delivery.

c. Event Driven Microservice Architecture:

i. Service Isolation & Autoscaling: Break down the Question Bank into smaller services for independent scaling. Automatically adjust resources based on traffic to handle spikes efficiently.
ii. Event-Driven Architecture: Use event-driven systems to decouple services and handle tasks asynchronously.

d. Load Balancing:

i. API Gateway and Load Balancer**:** Distribute requests across multiple instances for high availability.
ii. Global Load Balancing: Direct users to the nearest or least-loaded data center to reduce latency.

2. Recommend any architectural patterns or optimizations to maintain system efficiency.

To maintain system efficiency:

a. **Microservices Architecture**: Modularize the system into smaller, independent services that can scale individually.
b. **Caching Strategies**: Implement in-memory caching for frequently accessed data and use CDNs for static content.
c. **Asynchronous Processing**: Offload resource-intensive tasks to background jobs, reducing the load on real-time services.
d. **Load Balancing**: Distribute traffic evenly across service instances to prevent bottlenecks and ensure high availability.
e. **Read-Write Separation**: Optimize database operations by separating read and write requests across different instances.

These patterns and optimizations ensure that the system remains responsive and scalable as the user base grows.

3. Pseudo Code for Core Functions

Wrote a working prototype in python. Included as a .py file in the zip submission. Also available on my github repo: https://github.com/jayrajmulani/question-bank

Here is the code for all the core features:

```
def add_question(self, text, category, difficulty, tags=None):
    question_id = str(uuid.uuid4())  # Generate a unique identifier using UUID
    question = {
        "id": question_id,
        "text": text,
        "category": category,
        "difficulty": difficulty,
        "tags": tags if tags else []
    }
    self.questions.append(question)
    return question

def retrieve_questions(self, category=None, difficulty=None, tags=None):
    # Using list comprehension for optimized filtering
    return [
        question for question in self.questions
        if (category is None or question['category'] == category) and
           (difficulty is None or question['difficulty'] == difficulty) and
           (tags is None or any(tag in question['tags'] for tag in tags))
    ]

def search_by_keywords(self, keywords):
    # Using list comprehension for optimized keyword search
    return [
        question for question in self.questions
        if any(keyword.lower() in question['text'].lower() for keyword in keywords)
    ]
```

Part 2: Design Diagram