FRONT END BLOCK DATABASE

To Implement random question generation with Django, MySQL, XAMPP, and React.js, you'll need both frontend and backend components. Below, I'll provide you with some code snippets and steps for each part:

Backend (Django and MySQL using XAMPP):

1. Set Up Django Models:

In your Django models (models.py"), define a model for the

Backend (Django and MySQL using XAMPP):

Set Up Django Models:

In your Django models **("models.py"**), define a model for the questions.

```
from django.db import models

# Create your models here.

class Exam(models.Model):

    Question = models.CharField(max_length=100)

    Option1 = models.CharField(max_length=100)

    Option2 = models.CharField(max_length=100)

    Option3 = models.CharField(max_length=100)

    Option4 = models.CharField(max_length=100)

    Corrans = models.CharField(max_length=100)
```

2. Run Migrations:

After defining your models, run migrations to create the database

tables.

```
python manage.py makemigrations
```

python manage.py migrate

3. Create Django API Views:

Create Django views to handle API requests for random questions.

#views.py

```
from django.shortcuts import render
from django.http import JsonResponse
from rest_framework import serializers
from .models import Exam

# Define a serializer for the Exam model
class ExamSerializer(serializers.ModelSerializer):
    class Meta:
        model = Exam
        fields = '__all__'

def home(request):
    # Retrieve all Exam objects from the database
    exams = Exam.objects.all()

    # Serialize the Exam objects using the serializer
    serializer = ExamSerializer(exams, many=True)
    serialized_data = serializer.data

    # Return the serialized data as JSON response
    return JsonResponse({"exam": serialized_data}, safe=False)
```

Configure Django URLS:

Configure Django URLS ("urls.py") to map to your API views.

urls.py

```
from django.contrib import admin
from django.urls import path
from quiz.views import *

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',home)
]
# Add other URLS as needed
```

1.Frontend (React.js):

Fetch A // QuestionComponent.js

In your React component, use the 'fetch API to request random questions from your Django backend.

```
import React, { useEffect, useState } from 'react';

const QuestionComponent = () => {
    const [question, setQuestion] = useState({});

    useEffect(() => {
        fetch('http://localhost:8000/api/get_random_question/')
            .then(response => response.json())
            .then(data => setQuestion(data))
            .catch(error => console.error('Error fetching question:', error));
    }, []);
```

```
  return (
    <div>
      <h2>{question.text}</h2>
      {/* Render options and handle user interactions */}
    </div>
  );
};

export default QuestionComponent;
```

2. Integrate QuestionComponent:

Integrate your QuestionComponent into your main React application.

```
import React, { useState, useEffect } from 'react';

import logo from './logo.svg';
import './App.css';
import QuizApp from './quiz';
function App() {
 const [data, setData] = useState([]);

 useEffect(() => {
  // Fetch data from the Django API
  const fetchData = async () => {
   try {
    const response = await fetch('http://localhost:8000');
    const jsonData = await response.json();
    setData(jsonData);
   } catch (error) {
    console.error('Error fetching data:', error);
   }
```

```
    };


    fetchData();
  }, []);
  console.log(data)
  return (
    <div className="App">
    <QuizApp exam={data.exam}/>
    </div>
  );
}
export default App;
```

3. Run Your React App:

Run your React app to see the random question in action.

```
npm start
```

Ensure your Django server is running ('python manage.py runserver), and your MySQL database is set up and accessible.

4. API Endpoints

'http://localhost:8000'

Set the out Quiz.js Component to app.js

Note: Update the URLS and configurations based on your specific project structure and requirements. Adjust CORS settings in Django if your frontend and backend are on different domains. Additionally, handle user interactions and implement options rendering in your React component based on your specific question structure