QUIZ APP

1.Introduction

The purpose of this document is to outline the requirements for developing a random Question Generator application. The application will generate multiple choice questions with four answer choices. Users will select an answer, and the application will provide immediate feedback on whether the selected answer is correct or not. The questions will be randomized, ensuring that the same question is not repeated for a user during the test. The test will consist of five questions, and a summary of the results will be displayed at the end.

3.Project Overview

- Front End: React JS
- Back End: Django Python
- Database: MySQL

4.Backend Code

4.1 Create Django project and app:

Django-admin startproject quiz-project

cd quiz-project

python manage.py startapp quiz-app

4.2 Define Models**:**

from django.db import models

```
class Quiz(models.Model):
  title=models.CharField(max_length=255)
class Question(models.Model):
  quiz=models.ForeignKey(Quiz, on_delete=models.CASCADE)
  text=models.CharField(max_length=255)
class Choice(models.Model):
  questions=models.ForeignKey(Question_delete=models.CASCADE)
  text=models.CharField(max_length=255)
  is_correct=models.BooleanField(default=False)
```

4.3 Create Views:

```
from Django.shortcuts import render, get_object_or_404
from .models import Quiz

def quiz_list(request):
  quizzes=Quiz.objects.all()
  return render(request,'quiz_app/quiz_list.html', {'quizzes': quizzes})
def quiz_detail(request, quiz_id):
  quiz=get_object_or_404(Quiz, pk=quiz_id)
  return render(request, 'quiz_app/quiz_detail.html' , {'quiz': quiz)}
```

4.4 Html code for create template:

```
<u1>
    {%  for quiz in quizzes %}
        <li><a href="{% url 'quiz_app:quiz_detail' quiz.id%}">{{quiz.title}}</a></li>
    {% endfor %}
</u1>


<h1>{{ quiz.title }}</h1>
<u1>
   {% for question in quiz.question_set.all %}
    <li>{{ question.text }}</li>
    <u1>
        {% for choice in question.choice_set.all %}
            <li>{{ question.text }}</li>
        {% endfor %}
    </u1>
  {% endfor %}
</u1>
```

4.5 Configure URLs:

4.5.1 URL Pattern

```
from django.urls import path
from . import views
app_name='quiz_app'
urlpatterns=[
    path(' ', views.quiz_list, name='quiz_list'),
    path('<int:quiz_id>/ ', views.quiz_detail, name='quiz_detail'),
]
```

4.5.2 Main App URL:

```
from django.contrib import admin
from django.urls import path, include
urlpatterns=[
     path('admin/' , admin.site.urls),
     path('quiz/' , include('quiz_app.urls')),
]
```

4.6 Migration:

```
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
```

4.7 Run Development Server:

python manage.py runserver

5. Database Code:

-- Create a table to store quizzes

```
CREATE  TABLE quizzes (
     id SERIAL  PRIMARY  KEY,
     title VARCHAR(255) NOT NULL
);
```

--Create a table to store questions linked to quizzes

```
CREATE  TABLE  questions (
    id  SERIAL PRIMARY KEY,
    quiz_id INT NOT NULL,
    text VARCHAR(255) NOT NULL,
    FOREIGN KEY (quiz_id) REFERENCES quizzes(id) ON DELETE CASCADE
);
```

--Create a table to store choices linked in questions

```
CREATE  TABLE  choices(
    id  SERIAL PRIMARY KEY,
    question_id INT NOT NULL,
```

```
    text VARCHAR(255) NOT NULL,
    is_correct BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (question_id) REFERENCES questions(id) ON DELETE CASCADE
);
```

6. FrontEnd Code:

6.1 React Component:

```
// Quizlist.js
Import React, { useState, UseEffect } from 'react';

function Quizlist() {
    const [quizzes, setQuizzes]= useState([]);
    useEffect(() => {
        // Fetch quizzes from your backend API and set them in the state
        // Example: fetch('/api/quizzes').then()response =>response.json()).then
(data => setQuizzes(data));
    }, []);
    return (
        <div>
            <h1>Quiz List</h1>
            <u1>
                {quizzes.map(quiz => (
```

```
                        <li key={quiz.id}>
                            <a href={'/quiz/${quiz.id}'}>{quiz.title}</a>
                        </li>
                    ))}
                </u1>
            </div>
        );
}
export default QuizList;

// QuizDetail.js
Import React, { useState, UseEffect } from 'react';

function QuizDetail({ match }) {
    const [quiz, setQuiz]= useState({});
    const quizId = match.params.id;
    useEffect(() => {
        // Fetch quiz derails and questions from your backend API
        // Example: fetch('/api/quiz/${quizId}').then()response
=>response.json()).then        (data => setQuiz (data));
    }, [quizId]);
    return (
```

```
            <div>
                <h1>{quiz.title} </h1>
                 <u1>
                      {quiz.queations && quiz.questions.map(quiz => (
                         <li key={question.id}>
                             <p>{question.text}</p>
                             <u1>
                                {question.choices && question.choices.map(choice=> (
                            <li key={choice.id}>{choice.text}</li>
                       ))}
                  </u1>
            </div>
    );
}
export default QuizDetail;
```

6.2 Set React Router:

```
Import React from 'react';
Import {BrowserRouter as Router, Route, Switch} from 'react-router-dom';
Import QuizList from './QuizList';
Import QuizDetail from './QuizDetail';
```

```
function App() {
    return (
        <Router>
          <Switch>
            <Route path="/" exact component={QuizList}/>
             <Route path="/quiz/:id" component={QuizDetail}/>
           <Switch>
        <Router>
    );
}
export default App;
```

7. Application URL Link:

- The application URL link for the project is http://127.0.0.1:8000/quiz/.

8.Result Summary:

- In this project, at the end of the test the application will display a summary of the user's results.
- The summary included with the number of correct and incorrect answers.
- Using this , the user can see their score.

9.Conclusion:

This Project Requirement Document outlines the specifications and requirements for the development of the Random Question Generator application. By adhering to these requirements and utilizing the specified technologies , aim to deliver a MCQ test platform that generates randomized questions, provides immediate feedback, and displays a summary of the user's results at the end of the test.