Nama : I Gede Darma Ari Saputra

NIM : 2201010569

Kelas : C

Quiz 1

"EduConnect" adalah sebuah aplikasi mobile yang bertujuan untuk menghubungkan siswa, guru, dan orang tua dalam sebuah platform pendidikan terpadu. Aplikasi ini menyediakan fitur-fitur seperti kelas virtual, tugas online, ujian online, forum diskusi, kalender akademik, dan pelacakan kemajuan siswa.

Fitur Utama:

1. **Kelas Virtual**:
   - Guru dapat mengadakan kelas online dengan fitur video conferencing, sharing materi, dan whiteboard interaktif.
2. **Tugas Online**:
   - Guru dapat memberikan tugas dan siswa dapat mengumpulkan tugas mereka melalui aplikasi. Fitur ini juga dilengkapi dengan pengingat deadline.
3. **Ujian Online**:
   - Siswa dapat mengikuti ujian yang disediakan oleh guru dengan sistem penilaian otomatis.
4. **Forum Diskusi**:
   - Tempat bagi siswa, guru, dan orang tua untuk berdiskusi mengenai topik-topik pendidikan.
5. **Kalender Akademik**:
   - Menyediakan jadwal pelajaran, ujian, dan kegiatan sekolah lainnya.
6. **Pelacakan Kemajuan Siswa**:
   - Orang tua dan siswa dapat melihat laporan kemajuan akademik, termasuk nilai, kehadiran, dan feedback dari guru.

Alasan Menggunakan Teori Pendidikan Berbasis Teknologi:

1. **Aksesibilitas**:
   - Dengan teknologi, pendidikan menjadi lebih mudah diakses oleh siswa di berbagai lokasi, terutama di daerah terpencil.
2. **Interaktivitas dan Keterlibatan**:
   - Teknologi dapat membuat proses belajar mengajar menjadi lebih interaktif dan menarik, meningkatkan keterlibatan siswa.
3. **Penghematan Waktu dan Biaya**:
   - Mengurangi kebutuhan akan buku fisik dan alat tulis, serta menghemat waktu dengan tugas dan ujian online.
4. **Kolaborasi**:
   - Mendorong kolaborasi antara siswa, guru, dan orang tua melalui fitur diskusi dan pelacakan kemajuan.
5. **Pengelolaan Data yang Efektif**:

- Menggunakan database untuk menyimpan dan mengelola informasi akademik, memudahkan akses dan analisis data untuk meningkatkan kualitas pendidikan.

Teknologi yang Digunakan:

1. **Frontend**:
   - React Native untuk pengembangan aplikasi mobile yang cross-platform (iOS dan Android).
2. **Backend**:
   - Node.js dengan Express untuk membangun server dan API.
3. **Database**:
   - MongoDB untuk penyimpanan data yang fleksibel dan skala besar.
4. **Video Conferencing**:
   - Integrasi dengan layanan pihak ketiga seperti Zoom atau WebRTC.
5. **Authentication**:
   - Menggunakan Firebase Authentication untuk keamanan dan kemudahan dalam mengelola pengguna.

Dengan "EduConnect," diharapkan siswa, guru, dan orang tua dapat berinteraksi lebih efektif dan efisien dalam proses pendidikan, meningkatkan hasil belajar dan kesejahteraan akademik secara keseluruhan.

Main.Java

```
public class Main {

    public static void main(String[] args) {

        // Inisialisasi database

        Database db = new Database();

        db.connect();


        // Contoh membuat pengguna

        User teacher = new User("Guru", "guru@example.com",
"password", "teacher");

        db.addUser(teacher);


        User student = new User("Siswa", "siswa@example.com",
"password", "student");

        db.addUser(student);


        // Contoh membuat kelas virtual

        Classroom class1 = new Classroom("Matematika", teacher);
    db.addClassroom(class1);

        // Contoh membuat tugas
        Assignment assignment1 = new Assignment("Tugas 1", "Kerjakan
soal-soal", class1);
        db.addAssignment(assignment1);

        // Contoh melacak kemajuan
        ProgressTracker tracker = new ProgressTracker(student);
        tracker.addAssignment(assignment1, 90);
        db.addProgress(tracker);

        // Tutup koneksi database
        db.disconnect();
    }
```

User.Java

```
public class User {
    private String name;
    private String email;
    private String password;
    private String role; // "teacher" atau "student"

    public User(String name, String email, String password, String
role) {
        this.name = name;
        this.email = email;
        this.password = password;
        this.role = role;
    }
    // Getter dan setter
}
```

Classroom.Java

```
public class Classroom {
    private String name;
    private User teacher;

    public Classroom(String name, User teacher) {
        this.name = name;
        this.teacher = teacher;
    }

    // Getter dan setter
}
```

Assignment.Java

```
public class Assignment {
    private String title;
    private String description;
    private Classroom classroom;

    public Assignment(String title, String description, Classroom
classroom) {
        this.title = title;
        this.description = description;
        this.classroom = classroom;
    }

    // Getter dan setter
}
```

ProgressTracker.java

```java
import java.util.HashMap;
import java.util.Map;

public class ProgressTracker {
    private User student;
    private Map<Assignment, Integer> progress;

    public ProgressTracker(User student) {
        this.student = student;
        this.progress = new HashMap<>();
    }

    public void addAssignment(Assignment assignment, int score) {
        progress.put(assignment, score);
    }

    // Getter dan setter
}
```

Database.java

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class Database {
    private Connection connection;

    public void connect() {
        try {
            connection = 
DriverManager.getConnection("jdbc:sqlite:educonnect.db");
            System.out.println("Connected to database.");
            createTables();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    private void createTables() {
        String createUserTable = "CREATE TABLE IF NOT EXISTS users 
(id INTEGER PRIMARY KEY, name TEXT, email TEXT, password TEXT, role 
TEXT)";
        String createClassroomTable = "CREATE TABLE IF NOT EXISTS 
classrooms (id INTEGER PRIMARY KEY, name TEXT, teacher_id INTEGER)";
        String createAssignmentTable = "CREATE TABLE IF NOT EXISTS 
assignments (id INTEGER PRIMARY KEY, title TEXT, description TEXT, 
classroom_id INTEGER)";
        String createProgressTable = "CREATE TABLE IF NOT EXISTS 
progress (id INTEGER PRIMARY KEY, student_id INTEGER, assignment_id 
INTEGER, score INTEGER)";

        try (Statement stmt = connection.createStatement()) {
            stmt.execute(createUserTable);
            stmt.execute(createClassroomTable);
            stmt.execute(createAssignmentTable);
            stmt.execute(createProgressTable);
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    public void addUser(User user) {
        String sql = "INSERT INTO users (name, email, password, 
role) VALUES ('" + user.getName() + "', '" + user.getEmail() + "', 
'" + user.getPassword() + "', '" + user.getRole() + "')";

        executeUpdate(sql);

    }



    public void addClassroom(Classroom classroom) {

        String sql = "INSERT INTO classrooms (name, teacher_id) 
VALUES ('" + classroom.getName() + "', " + 
classroom.getTeacher().getId() + ")";

        executeUpdate(sql);

    }



    public void addAssignment(Assignment assignment) {

        String sql = "INSERT INTO assignments (title, description, 
classroom_id) VALUES ('" + assignment.getTitle() + "', '" + 
assignment.getDescription() + "', " + 
assignment.getClassroom().getId() + ")";

        executeUpdate(sql);

    }



    public void addProgress(ProgressTracker tracker) {

        for (Map.Entry<Assignment, Integer> entry : 
tracker.getProgress().entrySet()) {

            String sql = "INSERT INTO progress (student_id, 
assignment_id, score) VALUES (" + tracker.getStudent().getId() + ", 
" + entry.getKey().getId() + ", " + entry.getValue() + ")";

            executeUpdate(sql);

        }

    }



    private void executeUpdate(String sql) {

        try (Statement stmt = connection.createStatement()) {

            stmt.executeUpdate(sql);

        } catch (SQLException e) {

            System.out.println(e.getMessage());
        }
    }

    public void disconnect() {
        try {
            if (connection != null) {
                connection.close();
                System.out.println("Disconnected from database.");
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

QUIZ 2

```
import java.sql.*;
import java.util.HashMap;
import java.util.Map;

public class EduConnect {
    public static void main(String[] args) {
        // Inisialisasi database
        Database db = new Database();
        db.connect();

        // Contoh membuat pengguna
        User teacher = new User("Guru", "guru@example.com",
"password", "teacher");
        db.addUser(teacher);
        User student = new User("Siswa", "siswa@example.com",
"password", "student");
        db.addUser(student);

        // Contoh membuat kelas virtual
        Classroom class1 = new Classroom("Matematika", teacher);
        db.addClassroom(class1);

        // Contoh membuat tugas
        Assignment assignment1 = new Assignment("Tugas 1", "Kerjakan
soal-soal", class1);
        db.addAssignment(assignment1);

        // Contoh melacak kemajuan
        ProgressTracker tracker = new ProgressTracker(student);
        tracker.addAssignment(assignment1, 90);
        db.addProgress(tracker);

        // Tutup koneksi database
        db.disconnect();
    }
}

class User {
    private int id;
    private String name;
    private String email;
    private String password;
    private String role; // "teacher" atau "student"

    public User(String name, String email, String password, String
role) {
        this.name = name;
        this.email = email;
        this.password = password;
        this.role = role;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public String getEmail() {
        return email;
    }

    public String getPassword() {
        return password;
    }

    public String getRole() {
        return role;
    }
}

class Classroom {
    private int id;
    private String name;
```

```java
    private User teacher;

    public Classroom(String name, User teacher) {
        this.name = name;
        this.teacher = teacher;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public User getTeacher() {
        return teacher;
    }
}

class Assignment {
    private int id;
    private String title;
    private String description;
    private Classroom classroom;

    public Assignment(String title, String description, Classroom
classroom) {
        this.title = title;
        this.description = description;
        this.classroom = classroom;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public String getDescription() {
        return description;
    }

    public Classroom getClassroom() {
        return classroom;
    }
}

class ProgressTracker {
    private User student;
    private Map<Assignment, Integer> progress;

    public ProgressTracker(User student) {
        this.student = student;
        this.progress = new HashMap<>();
    }

    public void addAssignment(Assignment assignment, int score) {
        progress.put(assignment, score);
    }

    public User getStudent() {
        return student;
    }

    public Map<Assignment, Integer> getProgress() {
        return progress;
    }
}

class Database {
    private Connection connection;

    public void connect() {
        try {
            connection =
DriverManager.getConnection("jdbc:sqlite:educonnect.db");
            System.out.println("Connected to database.");
            createTables();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    private void createTables() {
        String createUserTable = "CREATE TABLE IF NOT EXISTS users
(id INTEGER PRIMARY KEY, name TEXT, email TEXT, password TEXT, role
TEXT)";
```

```
        String createClassroomTable = "CREATE TABLE IF NOT EXISTS
classrooms (id INTEGER PRIMARY KEY, name TEXT, teacher_id INTEGER)";

        String createAssignmentTable = "CREATE TABLE IF NOT EXISTS
assignments (id INTEGER PRIMARY KEY, title TEXT, description TEXT,
classroom_id INTEGER)";

        String createProgressTable = "CREATE TABLE IF NOT EXISTS
progress (id INTEGER PRIMARY KEY, student_id INTEGER, assignment_id
INTEGER, score INTEGER)";


        try (Statement stmt = connection.createStatement()) {

            stmt.execute(createUserTable);

            stmt.execute(createClassroomTable);

            stmt.execute(createAssignmentTable);

            stmt.execute(createProgressTable);

        } catch (SQLException e) {

            System.out.println(e.getMessage());

        }

    }


    public void addUser(User user) {

        String sql = "INSERT INTO users (name, email, password,
role) VALUES (?, ?, ?, ?)";

        try (PreparedStatement pstmt =
connection.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {

            pstmt.setString(1, user.getName());

            pstmt.setString(2, user.getEmail());

            pstmt.setString(3, user.getPassword());

            pstmt.setString(4, user.getRole());

            pstmt.executeUpdate();

            try (ResultSet generatedKeys = pstmt.getGeneratedKeys())
{

                if (generatedKeys.next()) {

                    user.setId(generatedKeys.getInt(1));

                }

            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    public void addClassroom(Classroom classroom) {
        String sql = "INSERT INTO classrooms (name, teacher_id)
VALUES (?, ?)";
        try (PreparedStatement pstmt =
connection.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {
            pstmt.setString(1, classroom.getName());
            pstmt.setInt(2, classroom.getTeacher().getId());
            pstmt.executeUpdate();
            try (ResultSet generatedKeys = pstmt.getGeneratedKeys())
{
                if (generatedKeys.next()) {
                    classroom.setId(generatedKeys.getInt(1));
                }
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    public void addAssignment(Assignment assignment) {
        String sql = "INSERT INTO assignments (title, description,
classroom_id) VALUES (?, ?, ?)";
        try (PreparedStatement pstmt =
connection.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {
            pstmt.setString(1, assignment.getTitle());
            pstmt.setString(2, assignment.getDescription());
            pstmt.setInt(3, assignment.getClassroom().getId());
            pstmt.executeUpdate();
            try (ResultSet generatedKeys = pstmt.getGeneratedKeys())
{
```

```
                if (generatedKeys.next()) {
                    assignment.setId(generatedKeys.getInt(1));
                }
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    public void addProgress(ProgressTracker tracker) {
        for (Map.Entry<Assignment, Integer> entry :
tracker.getProgress().entrySet()) {
            String sql = "INSERT INTO progress (student_id,
assignment_id, score) VALUES (?, ?, ?)";
            try (PreparedStatement pstmt =
connection.prepareStatement(sql)) {
                pstmt.setInt(1, tracker.getStudent().getId());
                pstmt.setInt(2, entry.getKey().getId());
                pstmt.setInt(3, entry.getValue());
                pstmt.executeUpdate();
            } catch (SQLException e) {
                System.out.println(e.getMessage());
            }
        }
    }

    public void disconnect() {
        try {
            if (connection != null) {
                connection.close();
                System.out.println("Disconnected from database.");
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
```

```
                    }
          }
}
```