KUISIONER PENILAIAN APLIKASI

Nama :

No.	Pertanyaan	Ya	Tidak
1.	Apakah aplikasi ini mudah digunakan?		
2.	Apakah aplikasi ini dapat membantu anak untuk mengenal nama-nama alat benda?		
3.	Apakah gambar dan penjelasan nama-nama alat musik mudah dimengerti?		
4.	Apakah dengan adanya aplikasi ini dapat membuat anak lebih mudah belajar?		
5.	Apakah aplikasi belajar ini bermanfaat?		

*Isilah kuisioner dengan memberi tanda *checklist* (√) pada kolom yang tersedia.

MASUKAN SISTEM (*Source Code*)

MainActivity.java

```
public class
MainActivity extends AppCompatActivity {
    Toolbar toolbar;
    RecyclerView recycler_category;
    MediaPlayer audio;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        audio = MediaPlayer.create(this, R.raw.sound);
        audio.setLooping(true);
        audio.start();
        toolbar = (Toolbar)findViewById(R.id.toolbar);
        toolbar.setTitle("QuizRul");
        setSupportActionBar(toolbar);
        recycler_category =
(RecyclerView)findViewById(R.id.recycler_category);
        recycler_category.setHasFixedSize(true);
        recycler_category.setLayoutManager(new
GridLayoutManager(this,2));
        //Get screen height
        //DisplayMetrics displayMetrics = new DisplayMetrics();
//getWindowManager().getDefaultDisplay().getMetrics(displayMetrics
);
        //int height = displayMetrics.heightPixels / 8; //Max size
of item in category
        CategoryAdapter adapter = new
CategoryAdapter(MainActivity.this,
DBHelper.getInstance(this).getAllCategories());
        int spaceInPixel = 4;
        recycler_category.addItemDecoration(new
SpaceDecoration(spaceInPixel));
        recycler_category.setAdapter(adapter);
    }
}
```

QuestionFragment.java

```
public class QuestionFragment extends Fragment implements
IQuestion {
    TextView txt_question_text;
    CheckBox ckbA, ckbB, ckbC, ckbD;
    FrameLayout layout_image;
    ProgressBar progressBar;
    Question question;
    int questionIndex =-1;
    public QuestionFragment() {
        // Required empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View itemView =
```

```
inflater.inflate(R.layout.fragment_question, container, false);
        //Get question
        questionIndex = getArguments().getInt("index",-1);
        question = Common.questionList.get(questionIndex);
        if (question != null) {
            layout_image = (FrameLayout)
itemView.findViewById(R.id.layout_image);
            progressBar =
(ProgressBar)itemView.findViewById(R.id.progress_bar);
            if (question.isImageQuestion())
            {
                ImageView img_question =
(ImageView)itemView.findViewById(R.id.img_question);
Picasso.get().load(question.getQuestionImage()).into(img_question,
new Callback() {
                    @Override
                    public void onSuccess() {
                        progressBar.setVisibility(View.GONE);
                    }
                    @Override
                    public void onError(Exception e) {
                        Toast.makeText(getContext(),
""+e.getMessage(), Toast.LENGTH_SHORT).show();
                    }
                });
            }
            else
                layout_image.setVisibility(View.GONE);
            //View
            txt_question_text = (TextView)
itemView.findViewById(R.id.txt_question_text);
            txt_question_text.setText(question.getQuestionText());
            ckbA = (CheckBox) itemView.findViewById(R.id.ckbA);
            ckbA.setText(question.getAnswerA());
            ckbA.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
                @Override
                public void onCheckedChanged(CompoundButton
buttonView, boolean b) {
                    if (b)
Common.selected_values.add(ckbA.getText().toString());
                    else
Common.selected_values.remove(ckbA.getText().toString());
                }
            });
            ckbB = (CheckBox) itemView.findViewById(R.id.ckbB);
            ckbB.setText(question.getAnswerB());
            ckbB.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
                @Override
                public void onCheckedChanged(CompoundButton
buttonView, boolean b) {
                    if (b)
Common.selected_values.add(ckbB.getText().toString());
                    else
Common.selected_values.remove(ckbB.getText().toString());
                }
            });
            ckbC = (CheckBox) itemView.findViewById(R.id.ckbC);
```

```
            ckbC.setText(question.getAnswerC());
            ckbC.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
                @Override
                public void onCheckedChanged(CompoundButton
buttonView, boolean b) {
                    if (b)
Common.selected_values.add(ckbC.getText().toString());
                    else
Common.selected_values.remove(ckbC.getText().toString());
                }
            });
            ckbD = (CheckBox) itemView.findViewById(R.id.ckbD);
            ckbD.setText(question.getAnswerD());
            ckbD.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
                @Override
                public void onCheckedChanged(CompoundButton
buttonView, boolean b) {
                    if (b)
Common.selected_values.add(ckbD.getText().toString());
                    else
Common.selected_values.remove(ckbD.getText().toString());
                }
            });
        }
        return itemView;
    }
    @Override
    public CurrentQuestion getSelectedAnswer() {
        //This function will return state of answer
        //Right, wrong or normal
        CurrentQuestion currentQuestion = new
CurrentQuestion(questionIndex,Common.ANSWER_TYPE.NO_ANSWER);
//Default no answer
        StringBuilder result = new StringBuilder();
        if (Common.selected_values.size() > 1)
        {
            //If multi choice
            //Split answer to array
            //Ex: arr[0] = A. Jakarta
            //Ex: arr[1] = B. Bandung
            Object[] arrayAnswer =
Common.selected_values.toArray();
            for (int i=0; i<arrayAnswer.length; i++)
                if (i<arrayAnswer.length-1)
                    result.append(new
StringBuilder(((String)arrayAnswer[i]).substring(0,1)).append(",")
); //take first letter of answer: Ex: arr[0] = A. jakarta, we will
take letter 'A'
                else
                    result.append(new
StringBuilder((String)arrayAnswer[i]).substring(0,1)); //Too
        }
        else if (Common.selected_values.size() == 1)
        {
            //If only one choice
            Object[] arrayAnswer =
Common.selected_values.toArray();
```

```
                result.append((String)arrayAnswer[0]).substring(0,1);
            }
            if (question != null)
            {
                //Compare correct answer with user answer
                if (!TextUtils.isEmpty(result)) {
                    if
(result.toString().equals(question.getCorrectAnswer()))
 currentQuestion.setType(Common.ANSWER_TYPE.RIGHT_ANSWER);
                    else
currentQuestion.setType(Common.ANSWER_TYPE.WRONG_ANSWER);
                }
                else
 currentQuestion.setType(Common.ANSWER_TYPE.NO_ANSWER);
            }
            else
            {
                Toast.makeText(getContext(), "Tidak dapat mendapatkan
soal", Toast.LENGTH_SHORT).show();
                currentQuestion.setType(Common.ANSWER_TYPE.NO_ANSWER);
            }
            Common.selected_values.clear(); //Always clear
selected_value when compare done
            return currentQuestion;
        }
        @Override
        public void showCorrectAnswer() {
            //Bold correct answer
            //Part : A,B
            String[] correctAnswer =
question.getCorrectAnswer().split(",");
            for (String answer:correctAnswer)
            {
                if (answer.equals("A"))
                {
                    ckbA.setTypeface(null,Typeface.BOLD);
                    ckbA.setTextColor(Color.RED);
                }
                else if (answer.equals("B"))
                {
                    ckbB.setTypeface(null,Typeface.BOLD);
                    ckbB.setTextColor(Color.RED);
                }
                else if (answer.equals("C"))
                {
                    ckbC.setTypeface(null,Typeface.BOLD);
                    ckbC.setTextColor(Color.RED);
                }
                else if (answer.equals("D"))
                {
                    ckbD.setTypeface(null,Typeface.BOLD);
                    ckbD.setTextColor(Color.RED);
                }
            }
        }
        @Override
        public void disableAnswer() {
            ckbA.setEnabled(false);
            ckbB.setEnabled(false);
```

```
            ckbC.setEnabled(false);
            ckbD.setEnabled(false);
        }
        @Override
        public void resetQuestion() {
            //Enable checkbox
            ckbA.setEnabled(true);
            ckbB.setEnabled(true);
            ckbC.setEnabled(true);
            ckbD.setEnabled(true);
            //Remove all selected
            ckbA.setChecked(false);
            ckbB.setChecked(false);
            ckbC.setChecked(false);
            ckbD.setChecked(false);
            //Remove all bold on text
            ckbA.setTypeface(null, Typeface.NORMAL);
            ckbA.setTextColor(Color.BLACK);
            ckbB.setTypeface(null, Typeface.NORMAL);
            ckbB.setTextColor(Color.BLACK);
            ckbC.setTypeface(null, Typeface.NORMAL);
            ckbC.setTextColor(Color.BLACK);
            ckbD.setTypeface(null, Typeface.NORMAL);
            ckbD.setTextColor(Color.BLACK);
        }
    }
```