
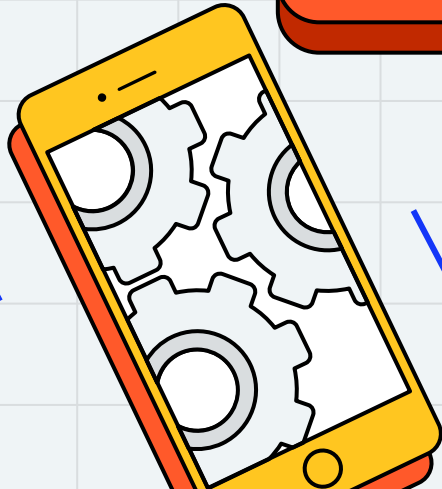


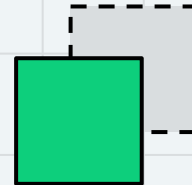


Application

Programming



Hend Alkittawi



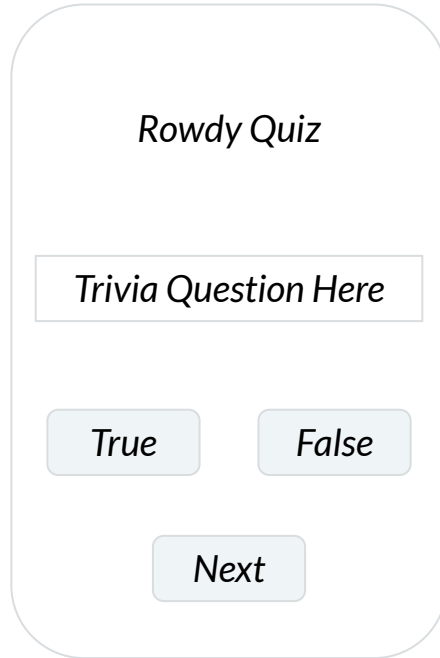


Android Development

Building Our First Android App

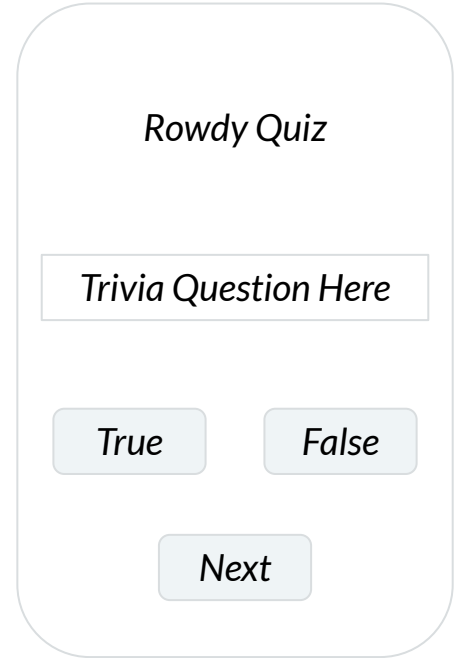
OUR FIRST ANDROID APPLICATION

- What do you think we need to build this app?

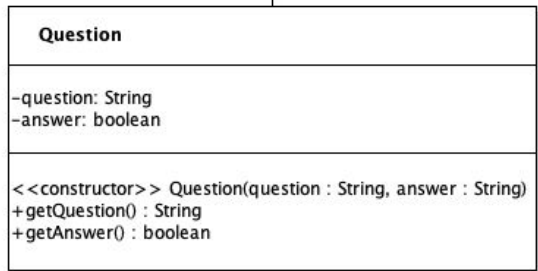
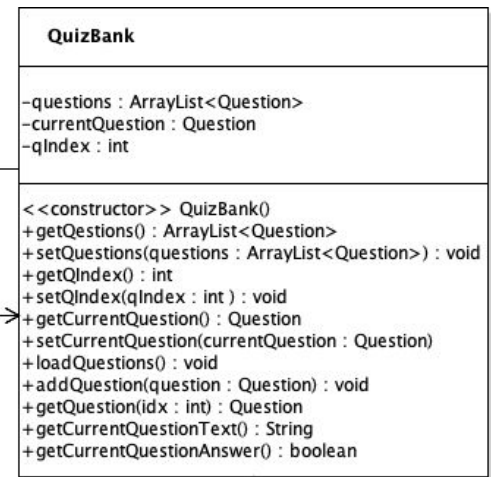
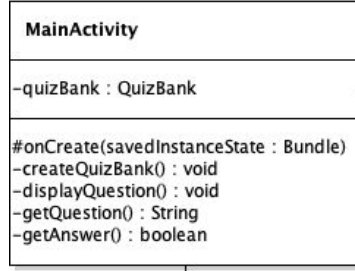
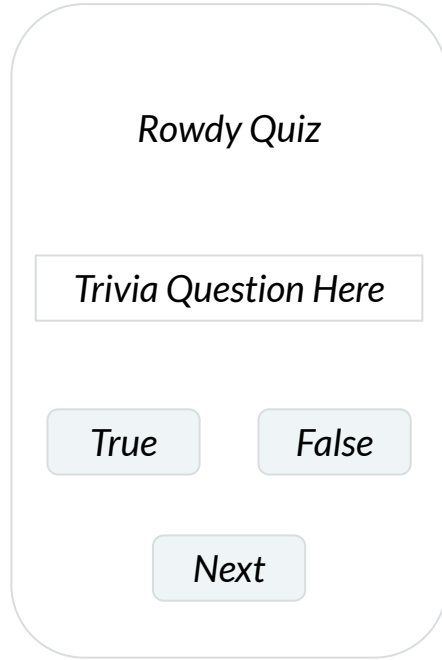


OUR FIRST ANDROID APPLICATION

- What do you think we need to build this app?
 - Model Objects
 - Question.java
 - QuizBank.java
 - View Objects
 - layout xml file
 - Controller Objects
 - MainActivity.java



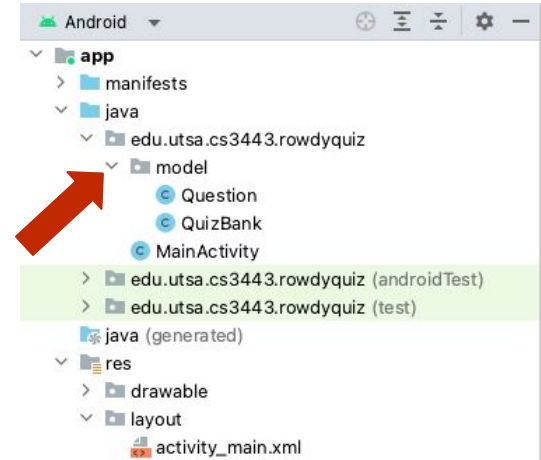
OUR FIRST ANDROID APPLICATION



OUR FIRST ANDROID APPLICATION

- Model Objects

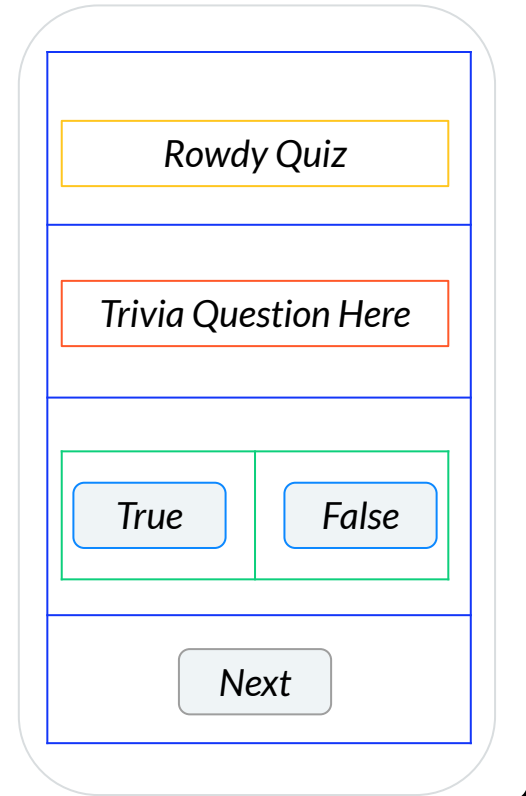
- After you create the project in Android Studio, **create a model package**
- In `edu.usta.cs3443.projectname`
 - right-click on the package
 - select New > Package
 - set the package name to **model**
 - create/place the model classes within the model package



OUR FIRST ANDROID APPLICATION

- View Objects

- layout xml file
- Set up the layout (xml or drag and drop)
 - **LinearLayout (Vertical)**
 - **TextView** - for Rowdy Quiz
 - **TextView** - for Trivia Question
 - **LinearLayout (Horizontal)**
 - **Button** - for True
 - **Button** - for False
 - **Button** - for Next



TIPS FOR WORKING WITH LAYOUT XML FILES

- Views may have an integer id associated with them. These ids are typically assigned in the layout XML files, and are used to find specific views within the view tree.
- Add **id attributes** for views that your code will be interacting with

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="24dp"
    android:text="trivia question here!"
    android:id="@+id/question_text"
/>
```


TIPS FOR WORKING WITH LAYOUT XML FILES

- Add strings to the string.xml resource file
- res > values > strings.xml

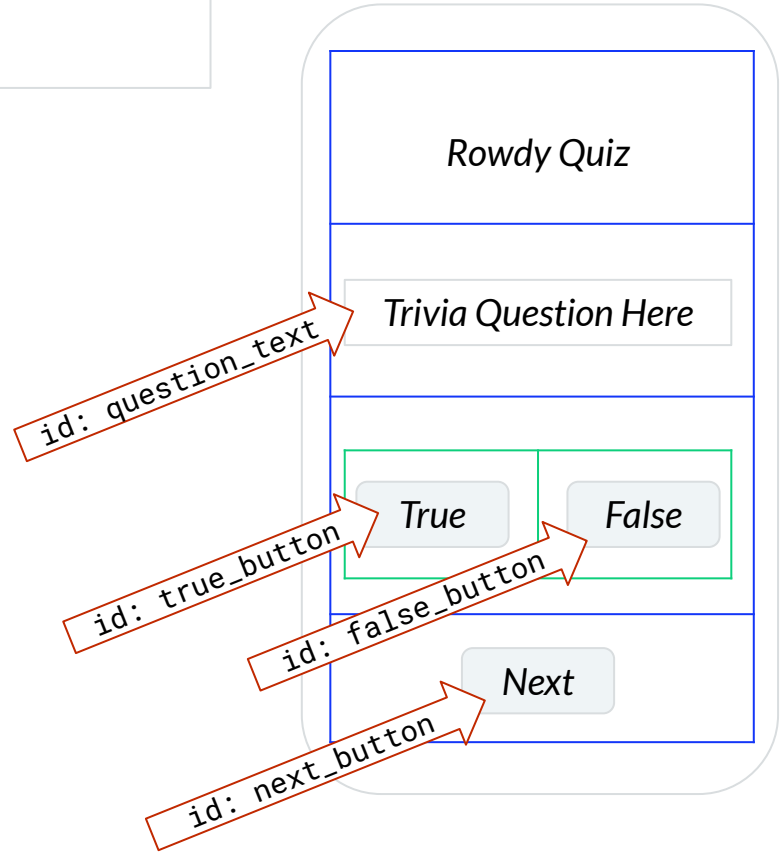
```
<resources>
    <string name="app_name">Rowdy Quiz</string>
    <string name="false_button">False</string>
    <string name="true_button">True</string>
    <string name="next_button">Next</string>
</resources>
```

- Use these strings with your views

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="24dp"
    android:text="@string/app_name"
/>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout width="match_parent"
    android:layout height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <TextView
        android:layout width="wrap_content"
        android:layout height="wrap_content"
        android:padding="24dp"
        android:text="@string/app_name" />
    <TextView
        android:id="@+id/question_text"
        android:layout width="wrap_content"
        android:layout height="wrap_content"
        android:padding="24dp"
        android:text="trivia question here!" />
    <LinearLayout
        android:layout width="wrap_content"
        android:layout height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:id="@+id/true_button"
            android:layout width="wrap_content"
            android:layout height="wrap_content"
            android:padding="24dp"
            android:text="@string/true_button" />
        <Button
            android:id="@+id/false_button"
            android:layout width="wrap_content"
            android:layout height="wrap_content"
            android:padding="24dp"
            android:text="@string/false_button" />
    </LinearLayout>
    <Button
        android:id="@+id/next_button"
        android:layout width="wrap_content"
        android:layout height="wrap_content"
        android:padding="24dp"
        android:text="@string/next_button" />
</LinearLayout>
```

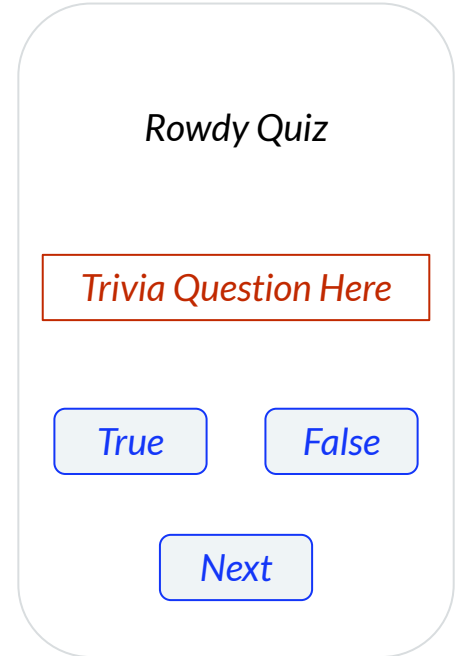
activity_main.xml



OUR FIRST ANDROID APPLICATION

- Controller Objects

- GUIs are built from **GUI components** called **views**
- A controller class **manages the flow of data** between the model layer and the view layer
- When the user interacts with GUIs, an **event object** is create
 - The *event object* is **dispatched** to an **event handler** (*listener*)
- A controller object **listens** and **responds** to these events



OUR FIRST ANDROID APPLICATION

- Controller Objects

- An Activity (controller) class utilizes the following to be able to **manage the flow of data, listen and respond** to events
 - **findViewById()** method to get references to the inflated View objects
 - **getId()** method to get the ID of a view
 - **OnClickListener interface** to set listeners on View objects to respond to user actions

- Rowdy Quiz

- Controller Objects

manage data flow

listen to events

respond to events

manage data flow

manage data flow

```
// some code is omitted, refer to full code on Github
public class MainActivity extends AppCompatActivity {
    private QuizBank quizBank;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        createQuizBank();
        displayQuestion();

        Button trueButton = findViewById(R.id.true_button);
        Button nextButton = findViewById(R.id.next_button);
        trueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(getAnswer())
                    // display a "Yay!" message
                else
                    // display a "Try again!" message
            } });
        nextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { displayQuestion(); }
        });
    }
    private void createQuizBank(){
        quizBank = new QuizBank();
        quizBank.loadQuestions();
    }

    private void displayQuestion(){
        quizBank.getCurrentQuestion();
        TextView questionText = findViewById(R.id.question_text);
        questionText.setText( getQuestion() ); }

    private String getQuestion(){
        return quizBank.getCurrentQuestionText(); }
    private boolean getAnswer(){
        return quizBank.getCurrentQuestionAnswer(); }
}
```

```

// some code is omitted, refer to full code on Github
public class MainActivity extends AppCompatActivity {
    private QuizBank quizBank;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        createQuizBank();
        displayQuestion();

        Button trueButton = findViewById(R.id.true_button);
        Button nextButton = findViewById(R.id.next_button);
        trueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(getAnswer())
                    // display a "Yay!" message
                else
                    // display a "Try again!" message
            } });
        nextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { displayQuestion(); }
        });
    }
    private void createQuizBank(){
        quizBank = new QuizBank();
        quizBank.loadQuestions();
    }
    private void displayQuestion(){
        quizBank.getCurrentQuestion();
        TextView questionText = findViewById(R.id.question_text);
        questionText.setText( getQuestion() );
    }

    private String getQuestion(){
        return quizBank.getCurrentQuestionText();
    }
    private boolean getAnswer(){
        return quizBank.getCurrentQuestionAnswer();
    }
}

```

listen to events

respond to events

manage data flow

manage data flow

Rowdy Quiz

Trivia Question Here

id: question_text

id: true_button

True

False

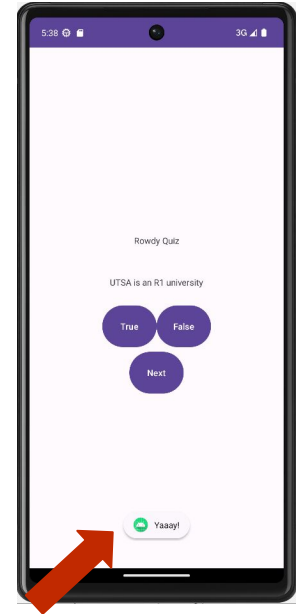
id: false_button

Next

id: next_button

OUR FIRST ANDROID APPLICATION

- Working with **Toasts**
 - a **Toast** is a pop-up message that informs the user of something but does not require any input or action
 - to create a Toast message use the `makeText()` method
 - the `show()` method shows the Toast view on the screen



```
Toast.makeText(view.getContext(), "Yaaay!", Toast.LENGTH_LONG).show();
```

```

// some code is omitted, refer to full code on Github
public class MainActivity extends AppCompatActivity {
    private QuizBank quizBank;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        createQuizBank();
        displayQuestion();

        Button trueButton = findViewById(R.id.true_button);
        Button nextButton = findViewById(R.id.next_button);
        trueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(getAnswer())
                    // display a "Yay!" message
                else
                    // display a "Try again!" message
            } });
        nextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { displayQuestion(); }
        });
    }
    private void createQuizBank(){
        quizBank = new QuizBank();
        quizBank.loadQuestions();
    }
    private void displayQuestion(){
        quizBank.getCurrentQuestion();
        TextView questionText = findViewById(R.id.question_text);
        questionText.setText( getQuestion() );
    }
    private String getQuestion(){
        return quizBank.getCurrentQuestionText();
    }
    private boolean getAnswer(){
        return quizBank.getCurrentQuestionAnswer();
    }
}

```

```

// some code is omitted, refer to full code on Github
public class QuizBank {
    private ArrayList<Question> questions;
    private int qIndex;
    private Question currentQuestion;

    public QuizBank(){
        questions = new ArrayList<Question>();
        qIndex = 0;
        currentQuestion = null;
    }

    public int getqIndex() { return qIndex; }
    public void setqIndex(int qIndex) { this.qIndex = qIndex; }

    public Question getCurrentQuestion(){
        if(getqIndex() >= 0 && getqIndex() < questions.size() ) {
            currentQuestion = questions.get(getqIndex());
            setqIndex(getqIndex() + 1);
        } else{
            setqIndex(0);
            currentQuestion = questions.get(getqIndex());
        }
        return currentQuestion;
    }

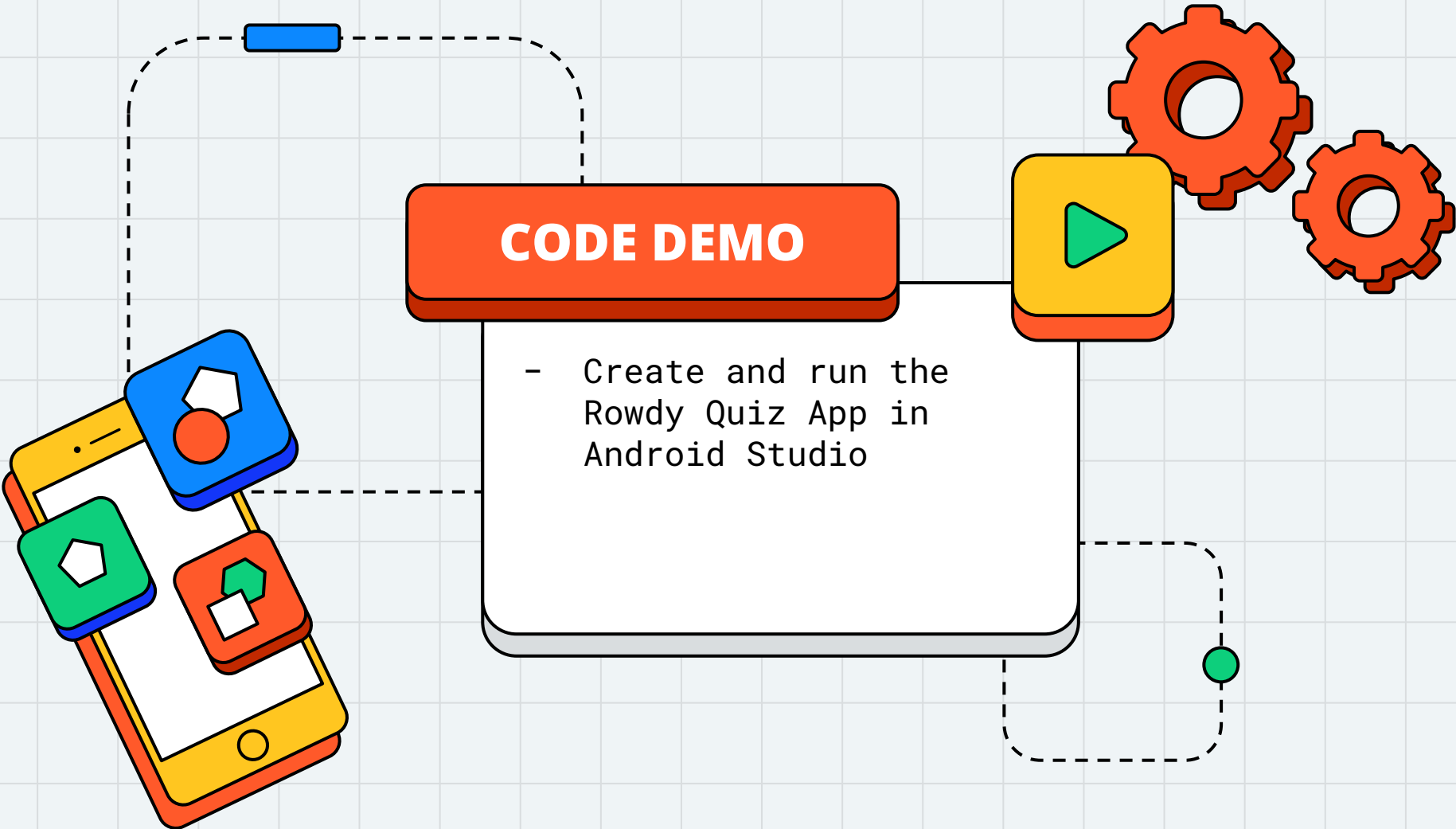
    public String getCurrentQuestionText(){
        return currentQuestion.getQuestion();
    }

    public boolean getCurrentQuestionAnswer(){
        return currentQuestion.getAnswer();
    }

    public void loadQuestions() {
        addQuestion(new Question("Is our mascot a roadrunner", true));
        addQuestion(new Question("Is UTSA in San Antonio", true));
    }

    public void addQuestion(Question question){
        questions.add(question);
    }
}

```

CODE DEMO

- Create and run the Rowdy Quiz App in Android Studio



THANK

YOU!



DO YOU HAVE ANY QUESTIONS?



hend.alkittawi@utsa.edu



By Appointment



Online