Penelope LTD

Software Development Contract

Recipient:
Sidharth Shanmugam

Author(s):
Giuseppe Barillari
Thomas Richardson

Reviewer(s):
Giuseppe Barillari
Alan Brookes
Ethan Cude
Harry Harrison
Oscar Goefron
Ophelia Korontini
Roman Kulev
Ana Monteiro
Thomas Richardson
Connall Shurey
Dawid Zaborowski

February 27, 2023

Contents

1 Introduction

This Software Development Agreement (referred to as the "Agreement" or "Software Development Agreement" throughout) states the terms and conditions that shall govern the contractual agreement between AppBar Ltd (the "Developer"), and Penelope LTD (the "Client"), who agrees to be bound by the terms of the Agreement.

The Client has listed the deliverables (the "Software") requirements in Exhibit A.
The Developer is a contractor with whom the Client has come to an agreement to develop the Software.

In consideration of the mutual covenants and promises made by both parties regarding this Agreement, the Developer and the Client (individually, a "Party", and collectively, "Parties") agree to the following terms:

2 Developer's duties

The Client hereby engages the Developer and the Developer agrees to be engaged by the Client to develop the Software in accordance with the specifications attached hereto as Exhibit A (the "Specifications").

1. The Developer shall complete the development of the Software according to the milestones described on the form attached hereto as Exhibit B. In accordance with such milestones, the final product shall be delivered to the Client by 17/03/23 (Friday, Spring Term Week 10).

2. For a period of 20 days after deliver of the final product, the Developer shall provide the Client with answers to any questions or assist in solving any problems with regard to the operation of the Software and agrees to respond to any reasonable request for assistance made by the Client regarding the Software within 20 days of the request.

3. Except as expressly provided in this Software Development Agreement, the Client shall not be obligated under this Agreement to provide any further support or assistance to the Developer.

4. The Client may terminate this Software Development Agreement at any time upon material breach of the terms herein and failure to resolve such a breach within 2 days of notification of such breach.

5. The Developer shall provide to the Client after the Delivery Date 50 hours of training with respect to the operation of the Software if requested by the Client.

3 Delivery

The Software shall function in accordance with the Specifications on or before the Delivery Date.

1. If the Software as delivered does not conform with the Specifications, the Client shall within 20 days of the Delivery Date notify the Developer in writing of the ways on which it does not conform with the Specifications. The Developer agrees that upon receiving such notice, it shall make reasonable efforts to correct any non-conformity.

2. The Client shall provide to the Developer written notice of its finding that the Software conforms to the Specifications within 20 days of the Delivery Date (the "Acceptance Date") unless it finds that the Software does not conform to the Specifications as described in Section 3.1 herein.

4 Compensation

In consideration for the Service, the Client shall pay the Company a fee of £240.00 (the "Compensation"), for all work under this Software Development Agreement.
The fee shall be paid by the Client in three separate instances:

25%	In advance
50%	Upon delivery
25%	Upon approval

If the Developer delivers the Software, functioning in accordance with the Specifications and approved by the Client, in advance up until the 10/03/23 (Friday, Spring Term Week 9), the Client shall provide an incentive in the form of an additional payment to the Developer of 15% of the Compensation.

5 Breach of Contract

If either party fails to deliver their respective obligations under this agreement, such party shall be in breach of this agreement. Upon a breach of this agreement by either party, the non-breaching party may seek any remedies available under law, including, but not limited to, damages for breach of contract or withholding further payments.
Breach of Contract includes but is not limited to: The Developer does not deliver the Software, functioning in accordance with the Specifications and approved by the Client before or on 17/03/23 (Friday, Spring Term Week 10).

6 Intellectual property rights in the software

The Parties acknowledge and agree that the Client will hold all intellectual property rights in the Software including, but not limited to, copyright and trademark rights. The Developer agrees not to claim any such ownership in the Software's intellectual property at any time prior to or after the completion and delivery of the Software to the Client.

7 Change in specifications

The Client may request that reasonable changes be made to the Specifications and tasks associated with the implementation of the Specifications. If the Client requests such a change, the Developer will use its best efforts to implement the requested change at no additional expense to the Client and without delaying delivery of the Software.
In the event that the proposed change will, in the sole discretion of the Developer, require a delay in the delivery of the Software or would result in additional expense to the Client, then the Client and the Developer shall confer and the Client may either withdraw the proposed change or require the Developer to deliver the Software with the proposed change and subject to negotiable delay and/or additional expense.
In the event that the Developer is not able to meet the new requirements the Client may withdraw the Agreement and any expense shall not be paid by the Client.

8 Confidentiality

The Developer shall not

(i) Disclose to any third party the business of the Client, details regarding the Software, including any information regarding the Software's code, the Specifications, or the Client's business (the "Confidential Information").

(ii) Make copies of any Confidential Information or any content based on the concepts contained within the Confidential Information for personal use or for distribution unless requested to do so by the Client.

(iii) Use Confidential Information other than solely for the benefit of the Client.

9 Developer warranties

The Developer represents and warrants to the Client the following:

1. Development and delivery of the Software under this Agreement are not in violation of any other agreement that the Developer has with another party.

2. The Software will not violate the intellectual property rights of any other party.

3. For a period of 20 days after the Delivery Date, the Software shall operate according to the Specifications. If the Software malfunctions or in any way does not operate according to the Specifications within that time, then the Developer shall take any reasonably necessary steps to fix the issue and ensure the Software operates according to the Specifications.

10 Idemnification

The Developer agrees to indemnify, defend, and protect the Client from and against all lawsuits and costs of every kind pertaining to the software including reasonable legal fees due to the Developer's infringement of the intellectual rights of any third party.

11 No modification unless in writing

No modification of this Agreement shall be valid unless in writing and agreed upon by both Parties.

12 Applicable law

This Software Development Agreement and the interpretation of its terms shall be governed by and construed in accordance with the laws of England and Wales and subject to the exclusive jurisdiction of the courts of England and Wales.

In witness whereof, each of the Parties has executed this Software Development Agreement, both Parties by its duly authorized officer, as of the day and year set forth below.

Penelope LTD	AppBar Ltd
Mr. Connall Shurey, Project Manager	Mr. Sidharth Shanmugam, Project Manager
Date	Date
Dr. Stuart Porter, Business Mentor	Professor Tony Ward, Business Mentor

13 Exhibit A

The Developer is asked to produce an Android Java Library (the "Library"). The Software is expected to be delivered in the form of source code. PENELOPE LTD requires APPBAR LTD to develop a Library to display text in accordance with the Software Engineering Project 2022/2023 standard previously agreed upon. The Library must contain:

- A class (the "Display class"), which extends the PresentationElement abstract class available in Appendix A, and implements the ViewElement interface available in Appendix B, to handle display of the provided information.
- Unit tests for the Display class.

The Developer must work on a GitHub repository provided by the Client.

13.1 The Display class

The Display class must provide a public constructor with the following signature:

```
public TextElement(String font, int fontSize, int color,
                   int x, int y, int width, int height,
                   long timeOnScreen)
```

Where:

- `font` can be `null`, `"mono"` or `"roboto"`. When `null`, the default `android.widget.TextView` font should be used. When `"mono"`, the font found at `R.font.chivo_mono_regular` should be used. When `"roboto"`, the font found at `R.font.roboto_condensed_regular` should be used.
- `fontSize` is to be interpreted in SP units.
- `color` is obtained using the standard Android Color utilities.
- `x` is in reference to the Client's coordinate system. Perform the following operation to obtain the x position on screen:

  ```
  int xPos = Math.round(
      (x * slide.getCalculatedWidth()) / (float) slide.getWidth()
  );
  ```

  where the `slide` object is provided in the `applyView` method.
- `y` is to be converted to pixels via the `dpToPx(int input)` provided method.
- `width` can be:

  - `MATCH_PARENT`, in which case the layout parameter for the element's width must be set to `RelativeLayout.LayoutParams.MATCH_PARENT`.
  - `WRAP_CONTENT`, in which case the layout parameter for the element's width must be set to `RelativeLayout.LayoutParams.WRAP_CONTENT`.
  - A value greater than 0, in which case the layout parameter for the element's width must be set to the result of the below opration:

```
Math.round(
    (width * slide.getCalculatedWidth()) / (float) slide.getWidth()
);
```

    where the `slide` object is provided in the `applyView` method.

- `height` can be:
  - `MATCH_PARENT`, in which case the layout parameter for the element's height must be set to `RelativeLayout.LayoutParams.MATCH_PARENT`.
  - `WRAP_CONTENT`, in which case the layout parameter for the element's height must be set to `RelativeLayout.LayoutParams.WRAP_CONTENT`.
  - A value greater than 0, in which case the layout parameter for the element's height must be set to the return value of the `dpToPx(int input)` provided method.
- `timeOnScreen` is in milliseconds, or `null` for a non-disappearing element.

The Display class must provide a setter method to set the text content, which takes a `String content` as parameter. The Display class must take all the above parameters in consideration to generate an `android.widget.TextView` (the "View"), identified by the existing ID `id` provided to the `applyView(View parent, ViewGroup container, Slide slide, int id)` method, upon invocation of aforementioned method.
The View is to be found within an `android.widget.RelativeLayout`.

13.2 Acceptance criteria

- The Display class is part of the package `com.penelope.faunafinder.presentation.elements`.
- The Display class has the appropriate constructor.
- The Display class extends the provided abstract class.
- The Display class implements the provided interface.
- The Unit tests effectively assert all the parameters are the expected ones.

- The `android.widget.TextView` parameters are set in accordance to the guidance provided above.
- The `android.widget.TextView` is rendered in accordance with the parameters set.
- The `android.widget.TextView` displays the provided text content.
- The library uses the third party Robolectric library for testing, if needed.
- No additional third party library is permitted.
- The Display class implementation returns the constant `TEXT_ELEMENT` upon invocation of the `getViewType()` method.
- The Display class implementation returns the text content, all in lowercase, upon invocation of the `getSearchableContent()` method.
- For versions of Android greater than or equal to `Q`, the text content is justified with `LineBreaker.JUSTIFICATION_MODE_INTER_WORD`.
- The software is delivered on time.

See Appendix A and Appendix B for any mentioned method or constant.

13.3 Example data

Considering a `Slide slide` with:

- `slide.getCalculatedWidth()` equal to 1440.
- `slide.getWidth()` equal to 1920.

On an `mdpi` device, for the input:

```
<text fontName="mono"
      fontSize="22"
      colour="#000000FF" xCoordinate="560" yCoordinate="30"
      width="1300" height="-5" timeOnScreen="5">Greylag Goose</text>
```

The `android.widget.TextView` is expected to have:

- A left margin of 420 pixels.
- A top margin of 30 pixels.
- Black text colour.
- A font size of 22 SP.
- A width of 975 pixels.

- A height of `RelativeLayout.LayoutParams.WRAP_CONTENT`.
- A display time of 5 seconds, after which the visibility of the `android.widget.TextView` must be set to `View.INVISIBLE`.
- Content "Greaylag Goose", without quotes.

14 Exhibit B

Milestone schedule

Date	Project milestone
17/02/23	Q&A. The Client make themselves available for the Developer to ask to clarify any unclear requirement.
24/02/23	First iteration. The Developer is expected to have their environment set-up and have a Display class able to find and manipulate an `android.widget.TextView` within an `android.widget.RelativeLayout`, regardless of the required parameters.
03/03/23	Q&A. The Client make themselves available for the Developer to ask to clarify any implementation question.
10/03/23	Second iteration. The Developer is expected to have a Display class able to find and manipulate an `android.widget.TextView` within an `android.widget.RelativeLayout`, taking into account the `fontSize`, `font`, `color`, `width` and `height` parameters.
17/03/23	The Developer is expected to deliver the Software. The Library must function in accordance to the guidance given above.

15 Appendix

A

```
package com.penelope.faunafinder.presentation.elements;

import android.content.res.Resources;
import android.util.DisplayMetrics;
import android.util.TypedValue;
import android.widget.RelativeLayout;

/**
 * <code>PresentationElement</code> is the abstract class which provides
 * all the methods and constants every presentation needs and must provide.
 */
public abstract class PresentationElement {
    // Constants
    protected static final DisplayMetrics displayMetrics
            = Resources.getSystem().getDisplayMetrics();
    protected static final int MATCH_X_CLIENT_SIDE = -1;
    protected static final int MATCH_WIDTH_CLIENT_SIDE = -1;
    protected static final int ALIGN_CENTER_OF_PARENT = -2;
    private static final int ALIGN_END_OF_PARENT = -3;
    protected static final int MATCH_PARENT = -4;
    protected static final int WRAP_CONTENT = -5;

    // View types
    public static final String AUDIO_ELEMENT = "audio";
    public static final String IMAGE_ELEMENT = "image";
    public static final String TEXT_ELEMENT = "text";
    public static final String VIDEO_ELEMENT = "video";


    // Requested screen position
    protected final int x, y;

    protected PresentationElement(int x, int y) {
        this.x = x;
        this.y = y;
    }

    /**
     * Method an element provides to inform about the its type
     *
     * @return {@link #AUDIO_ELEMENT} or {@link #IMAGE_ELEMENT} or
     * {@link #TEXT_ELEMENT} or {@link #VIDEO_ELEMENT}
     */
    abstract public String getViewType();
```

```
45
46      /**
47       * Method used by the adapter to filter search queries.
48       *
49       * @return The text content of an element,
50       * or null if no searchable content should be provided.
51       */
52      abstract public String getSearchableContent();
53
54      /**
55       * Utility used to convert DPs into pixels.
56       *
57       * @param input The value to convert in DPs.
58       * @return The converted value in pixels.
59       */
60      protected int dpToPx(int input) {
61          return Math.round(
62                  TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, input,
63                          displayMetrics));
64      }
65
66      /**
67       * Checks if layout rules are to be applied based on the requested screen position.
68       *
69       * @param layoutParams {@link RelativeLayout.LayoutParams} of the view being
70       *                     manipulated by the element.
71       * @return Boolean, whether no rules have been applied (true)
72       * or a rule has been applied (false).
73       */
74      protected boolean noHorizontalLayoutRulesToApply(
75              RelativeLayout.LayoutParams layoutParams) {
76          switch (x) {
77              case ALIGN_END_OF_PARENT:
78                  layoutParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
79                  return false;
80              case ALIGN_CENTER_OF_PARENT:
81                  layoutParams.addRule(RelativeLayout.CENTER_HORIZONTAL);
82                  return false;
83              default:
84                  return true;
85          }
86      }
87  }
```

B

```java
package com.penelope.faunafinder.presentation.elements;

import android.view.View;
import android.view.ViewGroup;

import com.penelope.faunafinder.xml.slide.Slide;

/**
 * Any class extending {@link PresentationElement} and handling
 * a class which extends {@link View} must implement this interface.
 */
public interface ViewElement {
    /**
     * Method called by the ViewHolder to apply view manipulations.
     *
     * @param parent    The View containing the element,
     *                  usually {@link android.widget.RelativeLayout}
     * @param container The ViewGroup attached to the ViewHolder.
     * @param slide     The slide the element belongs to.
     * @param id        The ID of the View the element is going to manipulate.
     * @return The manipulated View.
     */
    View applyView(View parent, ViewGroup container, Slide slide, int id);
}
```