Reading App Builder

Building Apps

Reading App Builder: Building Apps

Last updated: 17 July 2024

The latest version is available at
http://software.sil.org/readingappbuilder/resources/

and on the Help menu of Reading App Builder.

Contents

1. Preparing content for your app

Before you build an app with Reading App Builder (RAB), you need to get your content (text, images and audio) into formats that RAB can handle.

1.1. Preparing text

The text needs to be in one of the following formats:

1. **Word document (.docx)**

   RAB can import text and images from Microsoft Word (.docx) documents. This is the recommended format for text and is likely to be used by most users of RAB.

   When Word documents are displayed in the app, basic formatting will be preserved such as character styles (bold, italic, underline), numbered lists, bullet points, hyperlinks and simple tables.

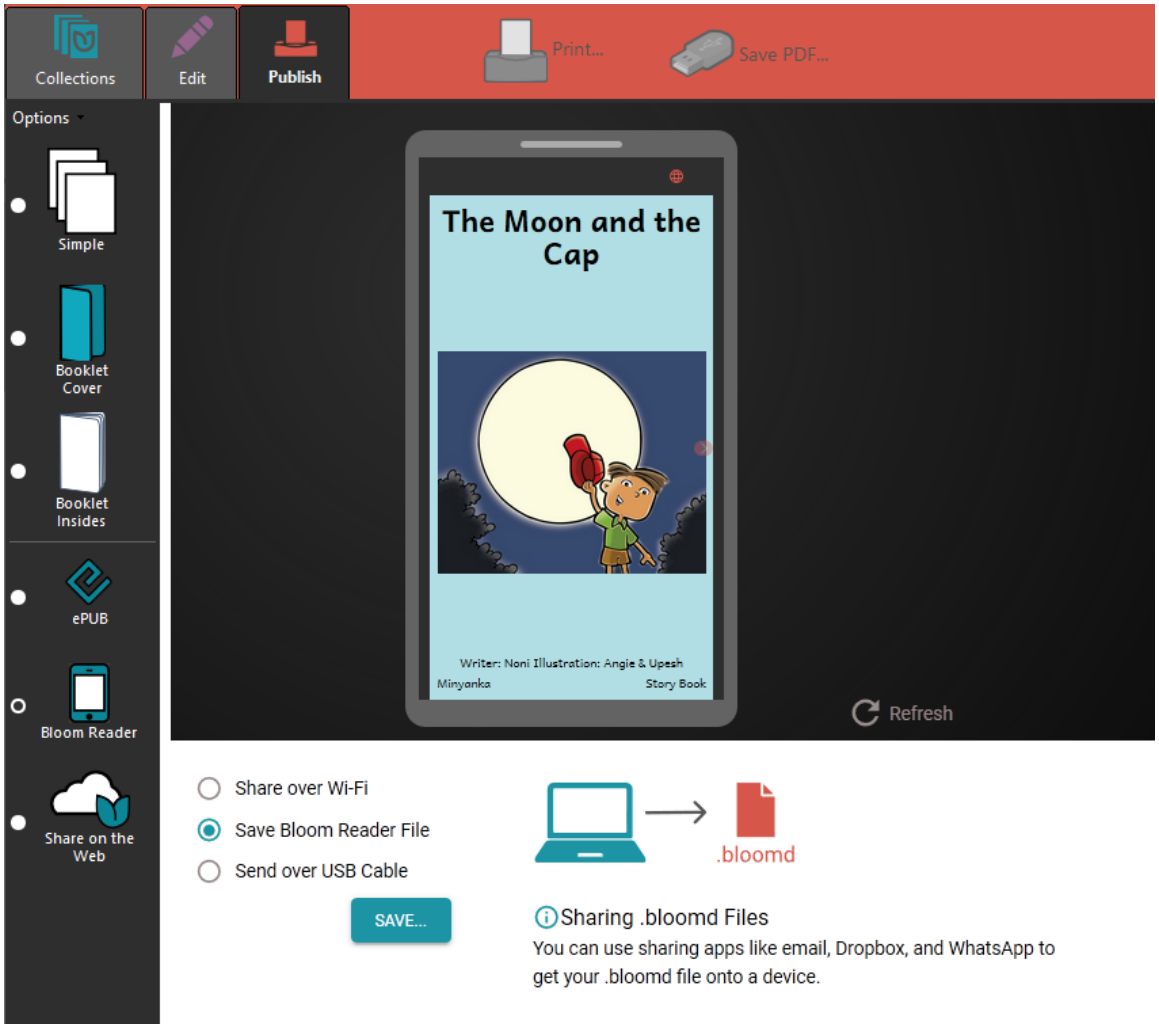
   To define separate chapters or pages, insert page breaks using CTRL+Enter.

2. **Bloom books (.bloompub)**

   RAB can create apps with books created with Bloom. It uses an embedded Bloom Player, as used in the Bloom Reader app.

   To create a Bloom Reader file for use in RAB:

   1. Open the Bloom book in **Bloom**.
   2. Select the **Publish** tab at the top of the Bloom screen.
   3. Select **Bloom Reader** on the left of the screen.
   4. Select the option **Save Bloom Reader File**.
   5. Click the **Save** button.
   6. Save the Bloom book to your computer as a **.bloompub** file. This is the file that you will need in Reading App Builder.



To find out more about Bloom, please see: http://bloomlibrary.org

3. SFM text files

RAB can import text from SFM (standard format marker) files. This is a good option for those who are familiar with Paratext and USFM markers.

In SFM text files, the chapters, section headings and paragraphs are marked by standard format markers such as \c, \s and \p. For more details, please see http://paratext.org/about/usfm.

The first marker in the file must be \id XYZ, where XYZ is a code you choose.

SFM book files must be **plain text files**. If you have Unicode characters, the text files should use UTF-8 encoding.

To create a text file in Windows, use a text editor such as Notepad.

To create a text file on a Mac, use TextEdit, remembering to choose Plain text files in the preferences because otherwise the default file type is RTF (which contains a lot of additional formatting codes).

4. **HTML files (.html)**

   RAB can display HTML files and make use of associated stylesheets (CSS) and images. Audio-text synchronization is not supported yet with this format.

5. **EPUB documents (.epub)**

   There is basic support for displaying the contents of EPUB documents, including images and associated styles. Audio and audio-text synchronization is not supported yet with this format.

1.2. Preparing images

Images are imported automatically from Word documents and Bloom books. They need to be specified separately if you use SFM format.

Images should be in JPEG or PNG format. Keep the image size small enough so that they display well on a small screen and will not make the app size too large. If your pictures are in a Word document, you can use the **Compress Pictures** tool in Word. Otherwise, RAB will allow you to resize the images after you have added them to the app project.

1.3. Preparing audio

If you want to include audio files in your app, these need to be in MP3 or 3GP audio format. Normally this should be one audio file per page or chapter. You can also include audio clips which are short audio files that are played when the user taps on a word, phrase or image.

If you have a picture story book, you can have a single audio file for the whole book or one audio file per page.

Keep the audio files at a size where the quality is good enough for a phone and where the file size is not too large.

2. How to build your first app

To build your first app with Reading App Builder:

1. Launch **Reading App Builder** from its icon on the desktop.

2. Click **New App** on the toolbar. The New App wizard will appear.

3. On the first page of the wizard, specify the **App Name**, such as “Dogon Stories”, “Supyiré Proverbs”, etc. This is the main title of your app and will be seen by the user. Do not include underscores or hard to understand abbreviations.

Click **Next** to move to the next page.

4. On the second page of the wizard, specify the **Package Name**, a dot-separated string which uniquely identifies your app.

   More details about choosing a good package name can be found in section *4.1. How should I choose the package name?*)

   Click **Next** to move to the next page.

5. On the **Books** page, select the books you want to add to the app.

   Click **Next** to move to the next page.

6. On the page of the wizard titled **Copyright and Licensing**, specify the copyright and licensing information that you would like to appear on the About box in the app. This includes the copyright owner for the text. Use the **Copyright Helper** wizard to help you.

   If you do not know what to put here, please ask the publishing department of your organisation for advice. They will want to make sure you get this right and do not simply make a guess as to what to include.

   Click **Next** to move to the next page.

7. On the **Contents Menu** page, choose whether or not you want to have items created for each book in the Contents menu.

   Click **Next** to move to the next page.

8. On the page of the wizard entitled **Fonts**, choose the font. You can either select from the given list of fonts or specify a different TrueType font file.

   Click **Next** to move to the next page.

9. On the page of the wizard titled **Font Handling**, you can select GeckoView if you know that the standard Android components will have trouble displaying the text correctly (e.g. if it is a complex script). More information on GeckoView can be found in the **Fonts** section of this document.

   Click **Next** to move to the next page.

10. On the page of the wizard titled **Color Scheme**, choose the color scheme for the app. The color you choose is the one that will be used for the main app bar. Individual colors for text, titles, links, backgrounds, etc. can be customised later.

    Click **Next** to move to the next page.

11. On the page of the wizard titled **Default Interface Language**, choose the language you want users to see when they first enter the app. This can be the current system language.

    Click **Next** to move to the next page.

12. On the page of the wizard titled **Interface Languages**, choose the app interface languages that the user can choose between.

    Click **Next** to move to the next page.

13. On the page of the wizard entitled **Icon**, choose the application launcher icon. You can select one of the images in the table or if you have your own PNG image files for the icon, click **Browse** and select them.

    Click **Next** to move to the next page.

14. On the page of the wizard entitled **Signing**, you need to specify the keystore and alias to use to sign the app. An app must be signed in this way so that it can installed on an Android device.

    If you do not already have a keystore file (which you are unlikely to have if this is your first time using the program):

    i. Click **Create KeyStore**.

    ii. Enter a new filename for the keystore, such as “keystore1” or something like that. Specify a password. Click **Next** to continue.

    iii. Enter an alias name for a key to create within your new keystore, such as “key”. Specify a password, which can be the same as the password you entered on the previous page. Click **Next** to continue.

    iv. On the **Certificate Issuer** page, provide details of your organisation in at least one of the fields. Click **Next** to continue.

    v. A new keystore will be created for you. Click **Close**.

15. Back on the **Signing** page of the New App wizard, you need to specify the keystore password, select the alias and enter the alias password (just as you entered them in the step above).

    Click **Next** to continue.

16. On the page of the wizard entitled **Project**, you can enter modify the project name and add an optional description of the app project. Neither of these will be visible to the user of your app. They are just for your own use and might help you distinguish between multiple app projects.

Click **Next** to continue. The New App wizard will close and the app definition will be added to the tree view on the left of the screen.

17. Take a look at each of the app configuration pages by selecting them in the tree view on the left. Look in each of the tabs on each page to verify that you have the settings you want. You can always go back to them later to change them if you find you need to make modifications to fonts, colors, styles, etc.

18. When you have finished configuring the app, click the **Build Android App** button on the toolbar at the top of the screen.

    If something is not configured correctly for the build to work, you will be notified of this.

19. A black command box will appear. Wait about a minute while the app is compiled.

    The first time the build process is run, the compiler needs to connect to the internet to download some files. After this, subsequent app builds will not require internet access. See **Tools ➢ Settings… ➢ Build Settings** to turn on offline mode after the first app build.

20. If the build succeeds, you will have a new APK file – the installation file for an Android app.

The next section describes how to copy this APK file to your phone and launch the app.

3. Installing the app on your phone

In the above section, you have seen how to compile an Android app. The result is an APK file, the installation file for an Android app. You now need to copy this APK file to your phone, install it and launch the app.

Here is how to do this:

1. Connect your Android phone to your computer using a **USB data cable**.

   (Sometimes you get cheap USB cables that can only charge a phone but cannot transfer data, so make sure you have the right kind of cable.)

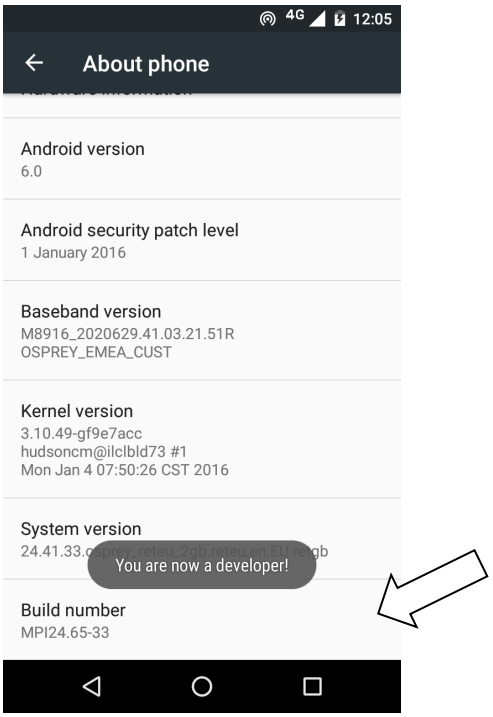
2. Ensure that **Developer Options ➢ USB Debugging** is enabled on your phone. By default, on new phones, Developer Options is turned off. This is how you can enable it:

   i. Open the **Settings** menu of your phone.

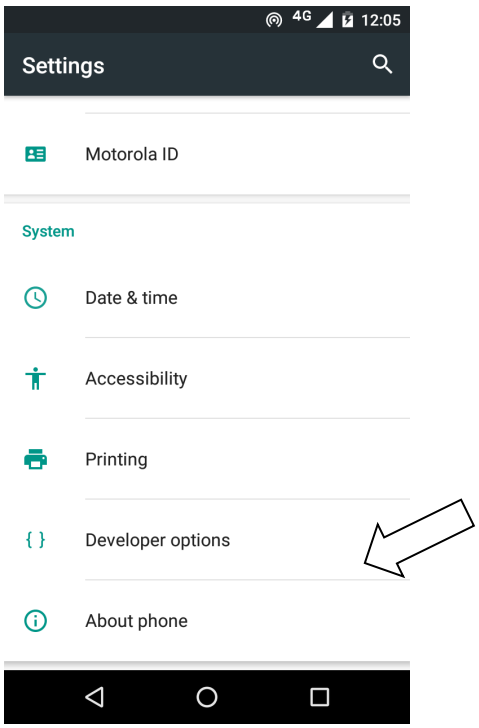
ii. Scroll down to the bottom of the menu and tap on **About Phone**.

iii. Find the **Build Number**. This could be on the About Phone page, or under a sub-menu such as 'Software Information'.

iv. Tap on the Build Number **seven times**. As you do this, you will see a series of messages appearing: "You are now 3 steps away from being a developer", "You are now 2 steps away from being a developer", "You are now 1 step away from being a developer", "You are now a developer!".

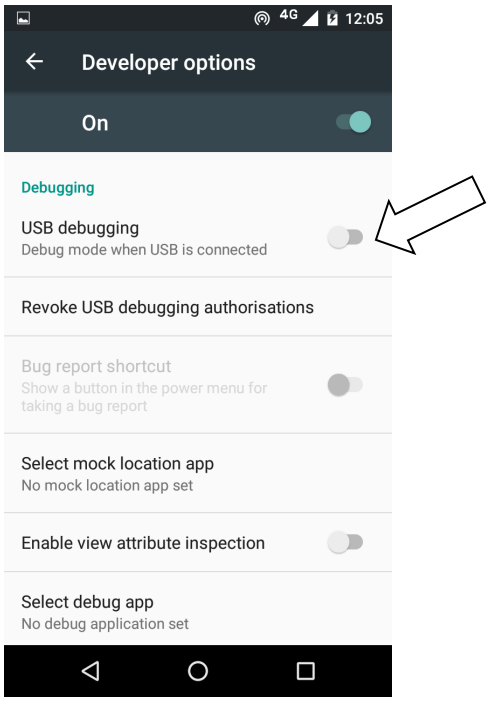


v. Now return to the Configuration menu of your phone. Look for the Developer Options menu item. You might see **Developer Options** above the **About Phone** menu item. If you do not see it here, it could be in **System** settings, under **Advanced**. Different phones place Developer Options in different places, so look around your Configuration menu until you find it.

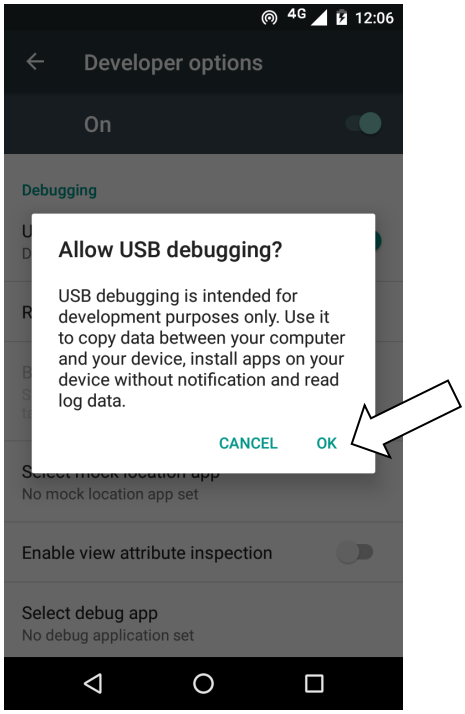


vi. Tap on **Developer Options** and ensure that it is turned on.

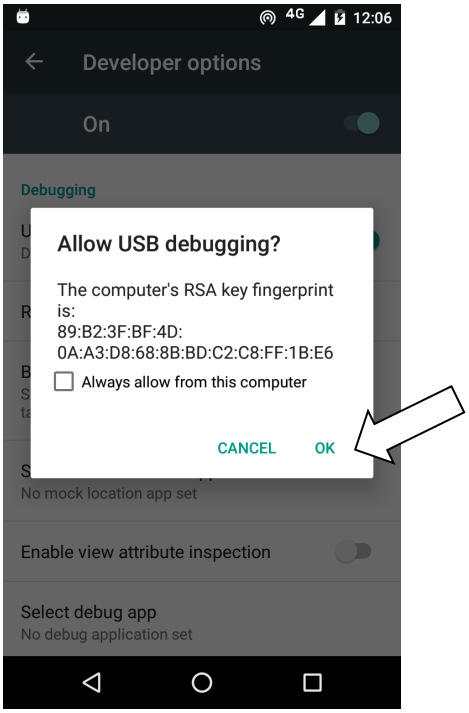
vii. Scroll down the Developer Options page and find **USB Debugging**. Enable this setting.



When you do this, you will probably get a message “Allow USB Debugging?”. Tap **OK**.



If you see a message box like this, tap **OK**:



3. In Reading App Builder, click the **Install APK** button on the toolbar at the top right of the screen.

   A command window will open and the APK file will be copied to your phone, installed and the app will be launched.

If this does not work, look at the command window to see if there is an error message. If you see a message such as “No devices/emulators found”, it means that your phone and computer are not connected correctly or that you have not enabled USB debugging on your phone.

Note:
Described above is a two-step process: **Build App** and then **Install APK**. If you prefer, you can tell Reading App Builder to do this in one step, i.e. for the APK to be installed and launched automatically after building an app. See **Tools ➢ Settings… ➢ After Build** to enable this feature.

4. App Creation Basics

4.1. How should I choose the app package name?

The standard for an app package name is to begin with the reversed web address of the publishing organisation, e.g. if it is SIL, the package name could begin with:

org.sil

and will be followed by something identifying the language and type of publication (Stories, Proverbs, Literacy, etc.), e.g.

org.sil.cccc.nnnn.stories

where ‘cccc’ is the country name and ‘nnnn’ is the language name.

If you work for a publishing organisation, you might have standards to follow for package names, so please contact your digital publications coordinator for advice on this.

Once you publish your app on an app store, you cannot change its package name later if you want users to continue to receive updates. The package name uniquely identifies the app in the Android world. Those who install the app will be able to find its package name on their device. It will also appear in the web address for your app if you make it available on Google Play.

If you are building apps for **test purposes** on your devices, you can use a package name beginning with com.example, e.g.

com.example.test.app123

But remember to change it before you publish the app.

4.2. Do I have to create a new keystore for each app, or can I reuse the same keystore for several of my apps?

You can use the same keystore and key alias for all or several of your apps.

See here for more details:
http://developer.android.com/tools/publishing/app-signing.html

4.3. I don't like the name "Reading App". Have you thought of calling the app something else?

The program that allows you to define and build apps is called "Reading App Builder" but the app itself doesn't have a name. It is up to you to choose the names for the apps you build.

You won't see 'Reading App' anywhere in the apps you create, so feel free to use an appropriate name in an international, national or local language.

4.4. Can I build apps when I do not have internet access?

The first time you build an app, you will need to be connected to the internet otherwise the compiler will fail. It will download a set of libraries used by the Gradle compiler. After that you can set the 'offline' version in **Settings** so you can work offline.

If you want to be able to build your first app without needing internet access, it is possible to copy the Gradle cache files from another computer that has already downloaded them. This will only work, however, if the absolute path to the files is exactly the same on the computer from which the files are taken as on your computer, e.g. "C:\gradle" on computer A and "C:\gradle" on computer B. It will not work if you have "C:\Users\John\.gradle" or computer A and "C:\Users\Jenny\.gradle" on computer B (which is the default Gradle cache folder).

So, on computer A, to get the cache files to distribute:

1. Go to **Tools** ➢ **Settings** ➢ **Build Settings** ➢ **Gradle Cache Folder**.
2. Enter "C:\gradle" and OK.
3. Build an app. The Gradle cache files will be downloaded to C:\gradle.

Then, on computer B:

4. Copy the C:\gradle folder from computer A to C:\gradle.
5. Go to **Tools** ➢ **Settings** ➢ **Build Settings** ➢ **Gradle Cache Folder**.
6. Enter "C:\gradle" and OK.

4.5. Can I build an app from the command line?

Yes, Reading App Builder has a command line interface which allows you to create a new app and build it, or load an existing app and build it.

The command line tool is named **rab** and can be found in the Program Files folder, usually c:\Program Files (x86)\SIL\Reading App Builder.

rab takes the following parameters:

Option	Description
`-new`	Create a new app project
`-load <project>`	Load an existing app project
`-build`	Build app project (use with either -new or -load)
`-no-save`	Do not save changes to app (use with -load)
`-?`	Show command line help
`-n <app-name>`	Set app name. Enclose the name in "double quotes" if it contains spaces.
`-p <package-name>`	Set package name, e.g. com.myorg.language.appname
`-b <filename>`	Add book or bundle file, such as a docx document
`-i <filename>`	Include additional parameters file. Use the full path of the file and enclose it in "double quotes" if there is a space in the path.
`-a <filename>`	Set about box text, contained in text file. Use the full path of the file and enclose it in "double quotes" if there is a space in the path.
`-f <fontname>`	Set font name or filename, e.g. "Charis SIL Compact", "c:\fonts\myfont.ttf" The font name must be one of the items in the list of fonts in the New App wizard. For other fonts, specify the full path to the font filename.
`-ic <filename>`	Add launcher icon (one or more .png files). Use the full path of the files and enclose them in "double quotes" if there is a space in the path.
`-l <lang-code>`	Set language for menu items and settings, e.g. en, fr, es
`-ft <feature=value>`	Set a feature, e.g. book-select=grid
`-vc <integer>`	Set version code, e.g. 1, 2, 3, or +1 to increment the current version code by 1.
`-vn <string>`	Set version name, e.g. 1.0, 2.1.4, or use +1, +0.1, +0.0.1 to increment the current value.
`-ks <filename>`	Set keystore filename. Use the full path of the file and enclose it in "double quotes" if there is a space in the path.

`-ksp <password>`	Set keystore password
`-ka <alias>`	Set key alias
`-kap <password>`	Set key alias password
`-fp <folder=path>`	Set a folder path, e.g. "app.builder=c:\Reading App Builder".

Examples:

```
rab -new -n \"My App\" -p com.example.myapp -b MyBookBundle.zip
          -f \"Charis SIL Compact\" -i keys.txt –build

rab -load \"My App\" -build
```

5. Fonts

If you are using a non-Roman script or a Roman script with combining diacritics, it is possible that Android devices will not display your fonts correctly. To overcome these problems, try using the GeckoView library.

GeckoView is a viewer component from Mozilla that replaces the standard Android viewer. It can render Graphite fonts correctly.

You can configure this on the **Appearance ➢ Fonts ➢ Font Handling** page.

The required GeckoView library files will add at least 55 MB to your app size, so do not enable GeckoView unless you know you need it to display your fonts correctly.

You will find that when you build an Android APK with GeckoView, the APK size will be at least 200 MB larger than without GeckoView. This is because it contains the GeckoView libraries for four different device architectures (32-bit ARM, 64-bit ARM, 32-bit Intel and 64-bit Intel). In practice, your app users do not need to install such a large file.

- For **online app distribution**, you need to upload an AAB file (app bundle) to Google Play rather than an APK. The AAB file contains all the GeckoView libraries for all four device architectures, but when a user installs an app from Google Play, Google will create a tailored APK for them, with just the libraries and components their phone needs. The AAB file might be over 300 MB, but the actual size of the app for any user will be much smaller.
- For **offline app distribution**, you can ask for Reading App Builder to create multiple APKs, one for each device architecture. Do this on the **App ➢ APK** tab. Each of these APKs will be significantly smaller (around 55 MB extra for GeckoView). You just need to choose the right one(s) to distribute, according to the types of phones people have. Many of the phones today will use the 64-bit ARM APK. Otherwise, the 32-bit ARM APK is used for most older phones. Intel

phones are less common, but it will depend on the phones being sold in your country.

6. Audio

6.1. How can I associate audio files with the text?

Reading App Builder allows you to use two different types of audio file:

- **Audio files that correspond to a page of text in the app**, e.g. the audio recording of a chapter of a book or the text in a picture story book. These can be linked to timing files to enable synchronised highlighting of the text when the audio is playing.
- **Short audio clips** which are played when the user taps a linked word, phrase or image in the app.

The first of these types of file is the most common in RAB apps and the instructions in this section will refer to these. For more information on using audio clips, please see section 6.5.

To associate audio with pages of text in your app, you will need one audio file per chapter. For example, if you have a book 'History of Mali' with 20 chapters, you will need 20 audio files, one for each chapter.

If you have several audio files per chapter or several chapters in one audio file, you will need to concatenate or split your files to create files of one chapter each before RAB can use them. This could be done in Audacity.

For **picture story books**, you can choose whether you have one audio file per page, or a single audio file for the whole book.

To add audio files:

1. Go to the **Audio Files** tab for the book, or the **Audio Files** tab for the book collection.
2. Click the **Add Audio Files…** button.
3. Select the audio files to add.

RAB will try to match the audio files to the chapters in the book(s), so you should use audio filenames that will make it obvious which file corresponds to which book and chapter.

The optional timing files can be added on the same page: one timing file per audio file.

6.2. How do I create the audio timing files for audio-text synchronization?

There are two possibilities:

1. Create the timing files manually using the **Audacity** audio editing program. For instructions, please refer to the document *Using Audacity for Audio-Text Synchronization*.

2. Automate the creation of the timing files using **aeneas**. For instructions, please refer to the document *Using aeneas for Audio-Text Synchronization*.

6.3. How do I distribute the audio files with the app?

There are three ways of including audio files in your app: assets, external folder or internet download. You can use a single **file source** for all of the files in an app or you can combine two or more sources in an app.

To specify the file source(s) in Reading App Builder, you need to visit the following two pages:

1. The **Media ➢ Audio ➢ File Source** page, which defines the available file sources.

   This page can be found in the apps tree view just under **Analytics** on the top level of app pages. You can modify, add and remove file sources here.

2. The **Audio Files** tab for a book or book collection, which lists the audio files with their corresponding file source.

   To change the source for a file or files, select the rows you want to change and select **Change Source**.

The follow sections describe the different file source types.

1. Assets

The audio files will be packaged inside the apk file for the app. This is the easiest method for a few files (e.g. one book) and requires no permissions. But be beware that the apk will get very large if you have several books of audio. The maximum size of an apk that can be uploaded to the Google Play store is 100 MB.

2. External Folder

No audio files are packaged within the app, so the apk is small. The app will look in a specified SD card folder to find the audio files it needs. If you are distributing the app via SD card, you include the folder of audio files on the SD card together with the apk. This method requires the 'Read external storage' permission but not internet access.

You can place the audio files inside sub-folders and sub-sub-folders in the specified SD card folder, using any folder names you choose. Alternatively, you can place all the audio files in a single folder without using any sub folders.

If the app does not find audio files in the specified folder or its sub-folders, it will also search the other folders on the device to see if it can find them there. For example, if the specified folder name is 'Audio 123' but the files are located in the 'Audio 456' folder instead, the app should find them. Once it has found a folder with a needed audio file, it will keep a note of it so it knows where to look next time.

3. Internet Download

Like method 2, no audio files are packaged within the app, so the apk is small. The app will look in a specified SD card folder to find the audio files it needs. If it doesn't find them there, it will look in all the other folders on the device. If it still cannot find them, the app can download the files one by one when it needs them from a website of your choice. This method requires the 'Read external storage', 'Write external storage', 'Connection state' and 'Internet' permissions.

Audio filenames

The internet download works best if your audio filenames do not include any spaces. A filename of the form "LNG-Story-01.mp3" is better than "LNG Story 01.mp3".

Http or https

The download manager in Android 2.3 (Gingerbread) cannot handle downloads from secure https addresses, so if you want to support these phones, use an http:// address instead of https://.

Audio file hosting

Recommended storage locations for the files on the internet include:

1. **A language-specific website**

   If you have a language-specific website for making resources available for download, you could place the audio files in a folder on the website.

   For example, if your website is called 'www.ourlanguage.org', you could upload the audio files to a folder called 'audio'. The http address for your audio files would then be: http://www.ourlanguage.org/audio

   Your website administrator should be able to help you do this.

2. **Cloud Storage Services**

   Amazon S3 (Simple Storage Service): http://aws.amazon.com/s3/
   Backblaze B2 Cloud Storage: https://www.backblaze.com/b2/cloud-storage.html
   Google Cloud Storage: https://cloud.google.com/storage/

   These cloud storage services are designed for fast, reliable and secure online storage. Once you have created an account, you create a 'bucket' in which to place your audio files. When you add the files, you need to make them public and make a note of the web address link to use to access them, e.g. http://s3.amazonaws.com/yourbucketname

You will get some months of free storage before there is a charge according to the bandwidth used, i.e. how many MB of audio users download. It might be easiest to organise this kind of cloud storage at an organisational level rather than creating a new account for each language.

6.4. How are the timing files distributed for the app?

If you have timing files for audio-text synchronisation, they are always packaged in the app assets whatever the audio source setting chosen in section 6.3 above. They are not downloaded from anywhere.

If timing files are not available when the app is compiled and are subsequently created by someone who isn't building the app (such as a volunteer), they can be placed in the external folder for testing. For more details, please see section 6.4 in the document *Using Audacity for Audio-Text Synchronization*, which answers the question *"I'd like to give someone else the task of creating timing files. Is there any way of them testing these without needing to know how to build the app?"*

6.5. How can I use audio clips in the app?

Audio clips are short audio files which are played when the user taps a linked word, phrase or image in the app.

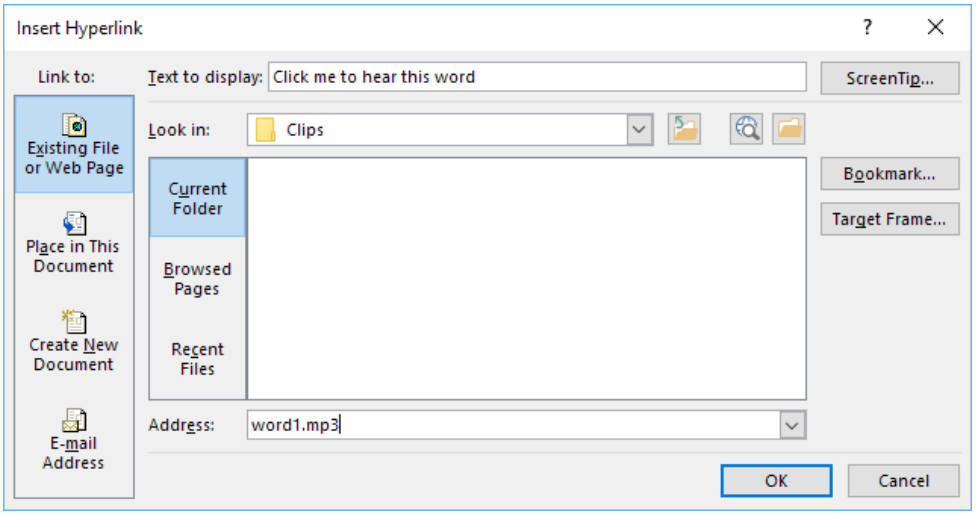
To include an audio clip in your app:

1. Select the **Audio** page for the app, and go to the **Audio Clips** tab.
2. Click **Add Audio Clips…** and select the audio file you want to add.
3. Add a hyperlink to this audio file in your text.

   You can do this using the markdown format with [] and (), e.g.

   [Click me to hear this word](word1.mp3)

   or if your text is in a Word document, select a word or a phrase, right-click, **Hyperlink…** and enter the audio filename as the address to link to.



Specify the audio filename without a path, e.g. “word1.mp3” rather than “C:\My clips\word1.mp3”.

7. Video

7.1. How do I include videos in the app?

The app can handle two types of video:

- **Videos hosted online for streaming**, e.g. YouTube, Vimeo, and MP4 streaming video files.
- **Videos for offline use**, MP4 files which can be packaged within the app assets, distributed on memory cards, or downloaded from the internet for use and sharing offline.

To add an **online** video:

1. Select the **Video** page in the app builder.
2. Click **Add Videos…**
3. Select **Online streaming video, from YouTube, Vimeo, etc.**, and click **Next**.
4. Enter the video URL in the Online Address field, e.g. http://www.youtube.com/watch?v=ABcDEdGHiJ.
5. Click **Finish**.
6. Double click on the video item in the table.
7. Specify a title (optional).
8. Select or modify the thumbnail image on the **Thumbnail** tab.

To add an **offline** MP4 video file:

1. Select the **Video** page in the app builder.

2. Click **Add Videos…**
3. Select **Video files in MP4 format**, and click **Next**.
4. Click **Add Videos…** and select the MP4 video files to add.
5. Click **Finish**.
6. Double-click on each video to edit its title (optional) and thumbnail image.

The next thing you need to do is tell the app builder where to insert and display the videos.

7.2. How do I reduce the size of a video that I want to package within the app?

If you want to include MP4 video files within the app, you need to make sure that they are as small as possible, otherwise your app will become too big to publish.

There are some good video compression tools online which can reduce the size of a video considerably without losing much quality. An example is the Clideo Compress Video tool, at https://clideo.com.

7.3. How do I specify where the videos will be displayed within the app?

There are two ways of inserting videos into the text:

- **Video markers in Word documents**: Use the word **VIDEO:**, together with the ID of the video within a book's Word document, e.g.

  This is a sub heading
  Here is an example video to view:
  VIDEO: V1

  Here is another video:
  VIDEO: V2

  The **VIDEO:** tag should be at the start of a new paragraph, and include a colon (:) before the video ID.

- **Video markers in SFM book files**: Use the **\video** marker, together with the ID of the video within an SFM book file, e.g.

  \s This is a sub heading
  \p Here is an example video to view:
  \video V1
  \p Here is another video:
  \video V2

8. About Page

Every app must have an About page, which is displayed to the user when they select **About** from the bottom of the app's navigation drawer menu.

To define the About page for your app:

1. Select **About** in the apps tree view on the left of the screen.
2. Enter the text to be displayed on the app's About page.
3. Select the **Viewer** tab to see how the text will appear in the app.

8.1. What information should I include in the About page?

The About page can contain any information. What you include is your choice, but commonly included items include:

- The app name and version
- The name of the organisation publishing the app
- Copyright and license information for the contents of the app (texts, images, audio). If you are using content from elsewhere, the license agreements will require you to include such information.
- Link to your website
- Link to a privacy policy
- Contact information, such as an email address or phone number.

8.2. Which formatting codes can I use in the About page?

You can use the following formatting codes:

Bold text	Surround the text you want in bold with and markers. Example: This is in bold
Italic text	Surround the text you want in bold with and markers. Example: This is in italics
Underlined text	Surround the text you want in bold with <u> and </u> markers. Example: <u>This is underlined</u>
Website links	Use the format [text](url), where 'text' is the text to display and 'url' is the web address. Example: Here is our [website](http://www.example.com)

Email links	Use the format [text](mailto:address), where 'text' is the text to display and 'mailto:address' contains the email address. Example: Contact us by [email](mailto:contact@example.com)
Phone numbers	Use the format [text](tel:number), where 'text' is the text to display and 'tel:number' contains the number to call. Example: Our number is [012-345-678](tel:012345678)
Image	To add an image, first add the image file to the app illustrations (**Images** ➢ **Illustrations**) and use the following HTML code: You can specify width information, as a percentage (e.g. "80%") or a fixed number of pixels (e.g. "80px"):
Left, right or centered alignment	To align text or images, surround them with the following markers: This is centered This is left aligned This is right aligned To center an image:
Styles	To apply styles (defined in the **Styles** page): This is styled as poetry
Fonts	To apply specific fonts to text, first make sure that the fonts are specified on the **Fonts** page. Then use the following syntax: This is in font1 This is in font2
Program Type	To display the program type (RAB), use the variable %program-type%
Program Version	To display the version of the program used to create the app, use the variable %program-version%

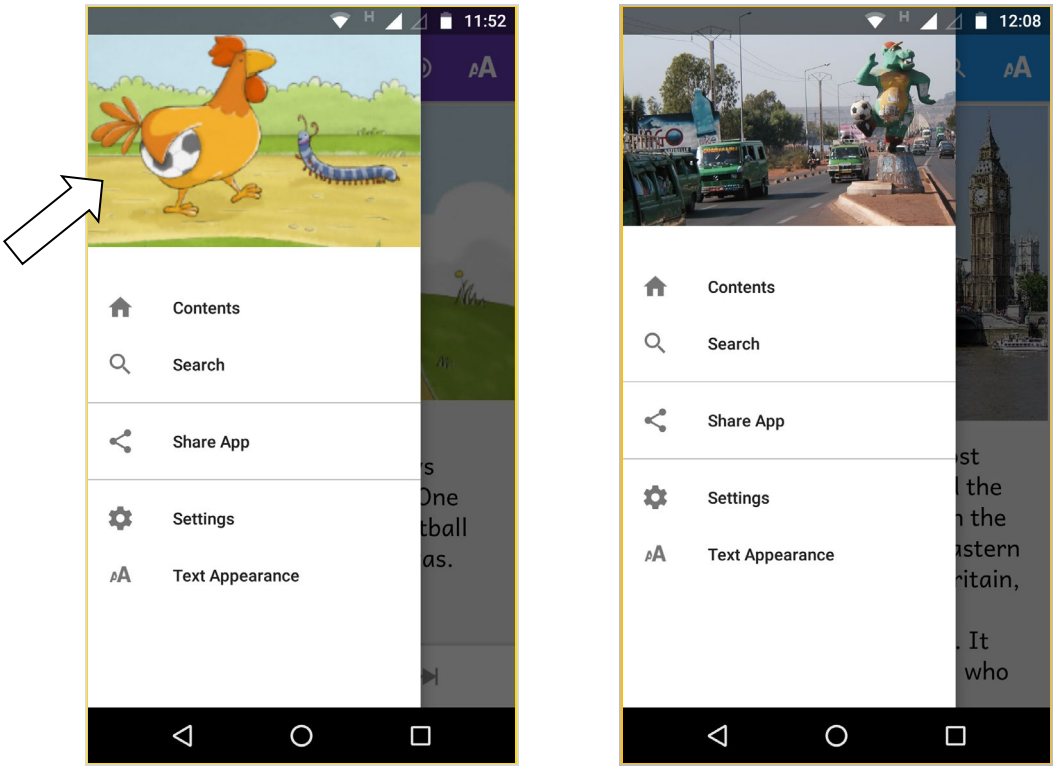
8.3. Which variables can I use in the About page?

You can use certain variables on the About page. These are replaced by text in the app. For example, the variable %version-name% will be replaced by the app version number, e.g. “1.2” in the app.

App name	To display the app name, use the variable %app-name%
App version	To display the app version, use the variable %version-name%
Program Type	To display the program type (SAB, RAB, DAB, KAB), use the variable %program-type%
Program Version	To display the version of the program used to create the app, use the variable %program-version%

9. Navigation Drawer

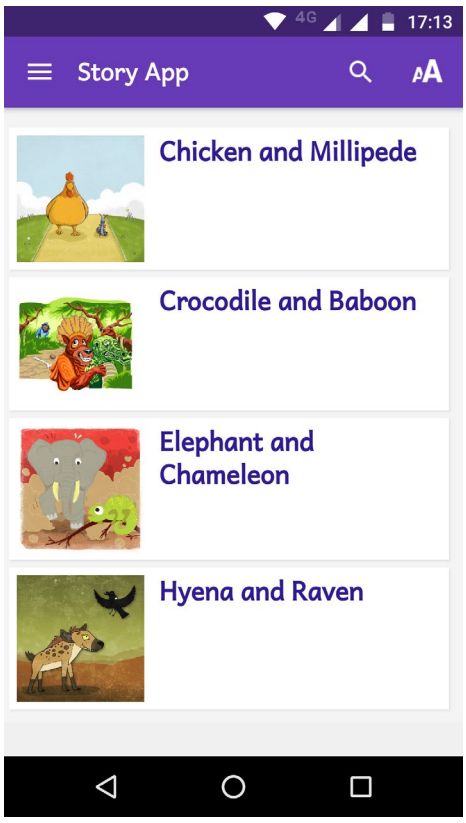
You can customise the image that appears at the top of the navigation drawer. It can be a photo, your organisation's logo or any relevant graphic design.



Specify a landscape image file on the **Appearance ➢ Graphics ➢ Navigation Drawer** page.

10. Contents Menu

When the user launches the app you will normally want to display a contents menu, giving them an easy way to select books in the app. The contents menu contains a list of images and titles.



10.1. How do I create a contents menu?

Items can be added automatically to the contents menu when you add books to the app (in the New App or Add Books wizards).

To add items to the contents menu:

1. Select the **Navigation** ➢ **Contents Menu** page in the Apps tree view on the left of the screen.

2. Ensure that the first tab, **Contents Items and Screens**, is visible.

3. For each contents menu item you want to add, click the **Add Item…** button and follow the wizard to add the item to the screen. You will need to specify:

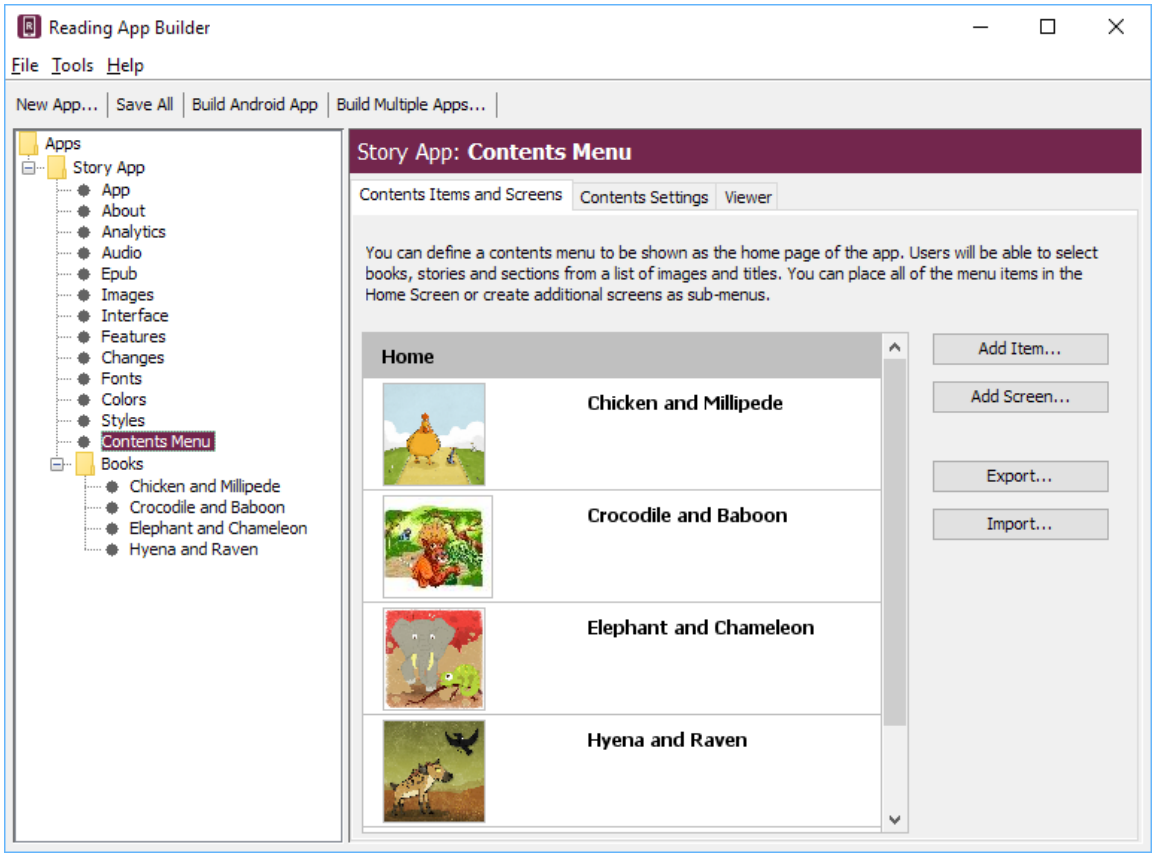
    - The **target reference**, such as a book - or a book and page number.
    - The **title** of the item, such as the book title
    - An optional **subtitle**

- An **image** (optional) – to appear to the left of the title.

4. You can reorder the items on the screen by selecting rows in the table and dragging up or down.

5. Go to the **Contents Settings** tab to specify display formatting and navigation options for the menu.

6. Select the **Viewer** tab to see how the contents menu will appear in the app.



10.2. How do I create additional contents menu screens?

In some cases you might want to have multiple contents screens, i.e. you select an item in the first screen and it takes you to a second contents screen. This is especially useful if you have a lot of books that you want to divide into categories.

To add a contents screen:

1. Click the **Add Screen…** button and give the screen a name.

2. Click the **Add Item…** button and specify its target to be **Another menu screen** and select the name of your new screen. This will mean that when the user taps on this item in the first screen they will be taken to your new screen.

3. Add menu items to your new screen as described above.

4. You can move items between screens by dragging item rows up or down.

11. Radio

You can include streaming web radio (internet radio) stations in your app.

To add a web radio station:

1. Go to **Media** ➢ **Radio**.

2. Click **Add Radio Station…**

3. Enter a unique ID for the station (not seen by the user), a short name, and the web address (URL) from which the radio station is being streamed. Sample URLs can be found at https://streamurl.link.

   Click **Next**.

4. Choose an image to display when the radio station is playing. This could be the radio station's logo.

   Click **Next**.

5. Choose whether to add a link to this station from the Contents menu.

   Click **Finish**.

Compile the app to test the radio stations on your device.

12. Picture Story Books

A picture story book is a book with a picture at the top of each page and a few sentences of text underneath. In the app, the picture will stay at the top of the screen and the text will scroll.

Audio-text highlighting works as for other kinds of book, and when the audio reaches the end of a page the app will move to the next page of the story.

When the device screen is in landscape orientation, the picture will be displayed to take up the whole screen. If you specify a timing file alongside the audio file, each phrase will be displayed as a subtitle over the image.

You can choose to associate a single audio file with the whole picture story book, or one audio file per page.

12.1. How do I define a picture story book?

Picture story books can be defined in one of three formats:

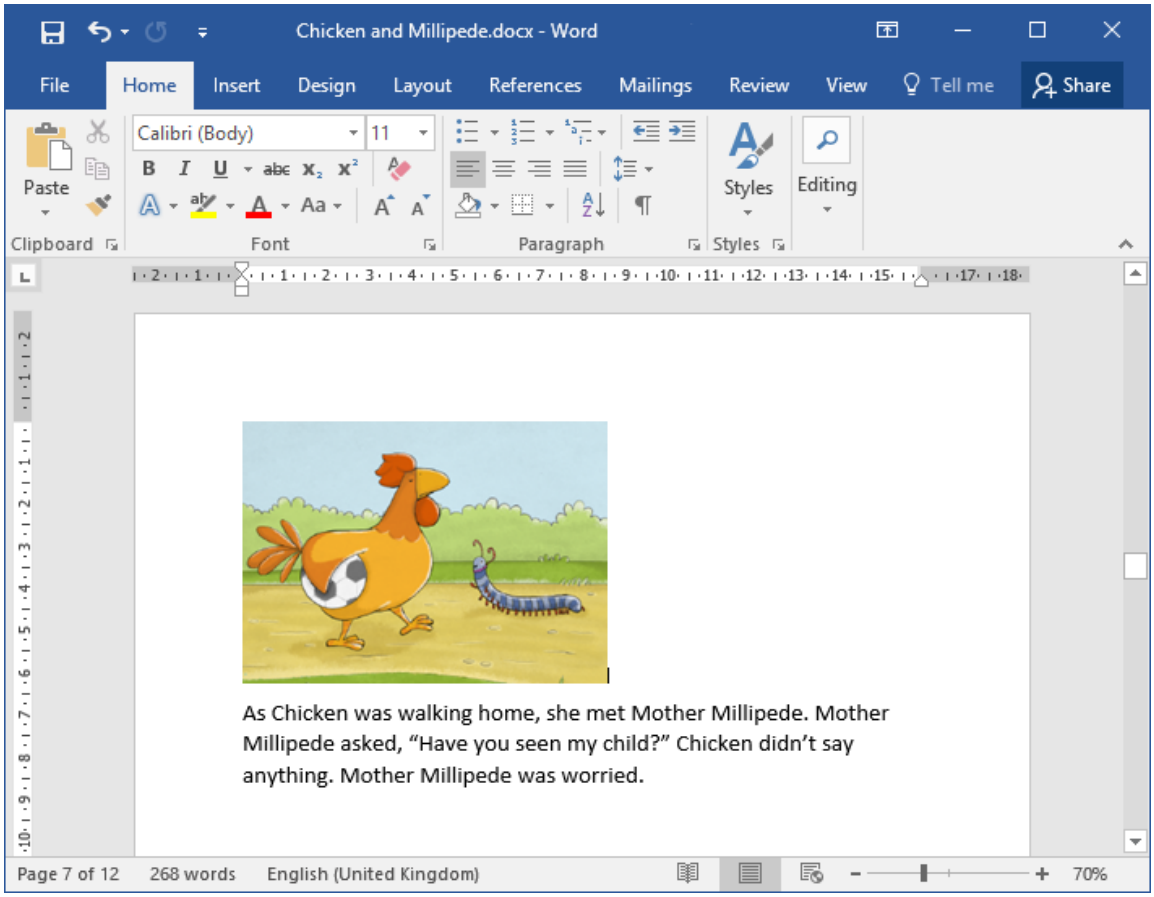
- Microsoft Word (.docx) documents,
- Bloom books, or
- Text files using standard format markers (SFM).

See 11.2 and 11.3 for a description of these formats.

12.2. What do picture story books in Word documents look like?

To create a Microsoft Word document to define a picture story book:

1. Start with a new blank document in Word.
2. Insert a picture at the top of the first page.
3. Add the story text for this page below the image.

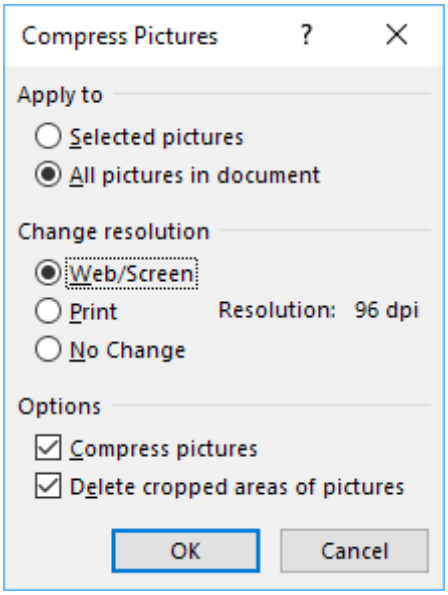


4. Insert a page break after the text using **CTRL+Enter**, or **Insert ➢ Page Break**.
5. Insert a picture at the top of the second page and place the story text under it.
6. Insert a page break after the text using **CTRL+Enter**, or **Insert ➢ Page Break**.
7. Continue adding additional pages in this way.
8. Save the document in **.docx** format.

You can add images to the document in .jpeg, .jpg, .png, .tif or .gif formats and Microsoft Word will convert them automatically to the format recognised by Reading App Builder.

Pictures are not compressed within the app at present, so if you have large image files it is best to resize them. You can do this either before adding them to the document or within Word itself:

1. Select one of the images in Word.
2. Click **Compress Pictures** on the picture format ribbon or toolbar.
3. Select **All Pictures in document**.
4. Select the resolution as **Web/Screen**.



5. Click **OK**.

6. Save the document.

12.3. What do picture story books in SFM format text files look like?

They start with an id of your choice:

```
\id CHICKEN
```

and a title:

```
\toc2 Chicken and Millipede
```

Each page begins with the \page marker:

```
\page 1
```

followed by an image filename:

```
\img chicken-01.jpg
```

and then the text for that page, using USFM markers like \p for indented paragraphs or \m for unindented paragraphs:

```
\m Chicken and Millipede were friends. But they were always
competing with each other. One day they decided to play football
to see who the best player was.
```

Here is an example:[1]

```
-----------------------------------
\id CHICKEN
\toc2 Chicken and Millipede

\page 1
\img chicken-01.jpg
\m Chicken and Millipede were friends. But they were always competing with each other. One day they decided to play football to see who the best player was.

\page 2
\img chicken-02.jpg
\m They went to the football field and started their game. Chicken was fast, but Millipede was faster. Chicken kicked far, but Millipede kicked further. Chicken started to feel grumpy.

\page 3
\img chicken-03.jpg
\m They decided to play a penalty shoot-out. First Millipede was goal keeper. Chicken scored only one goal.
\m Then it was the chicken's turn to defend the goal.

\page 4
\img chicken-04.jpg
\m Millipede kicked the ball and scored. Millipede dribbled the ball and scored. Millipede headed the ball and scored. Millipede scored five goals.
```

etc.

The book files must be **plain text files**. If you have Unicode characters, the text files should use **UTF-8 encoding**. To create a text file in Windows, use Notepad. To create a text file on a Mac, use TextEdit, remembering to choose Plain text files in the preferences because otherwise the default file type is RTF (which contains a lot of additional formatting codes).

12.4. Where do the pictures go?

If you have defined the picture story book as a Word document, the images in the document will be added automatically to the app illustrations collection when you add the book. You should see them on the **Illustrations** page for the book.

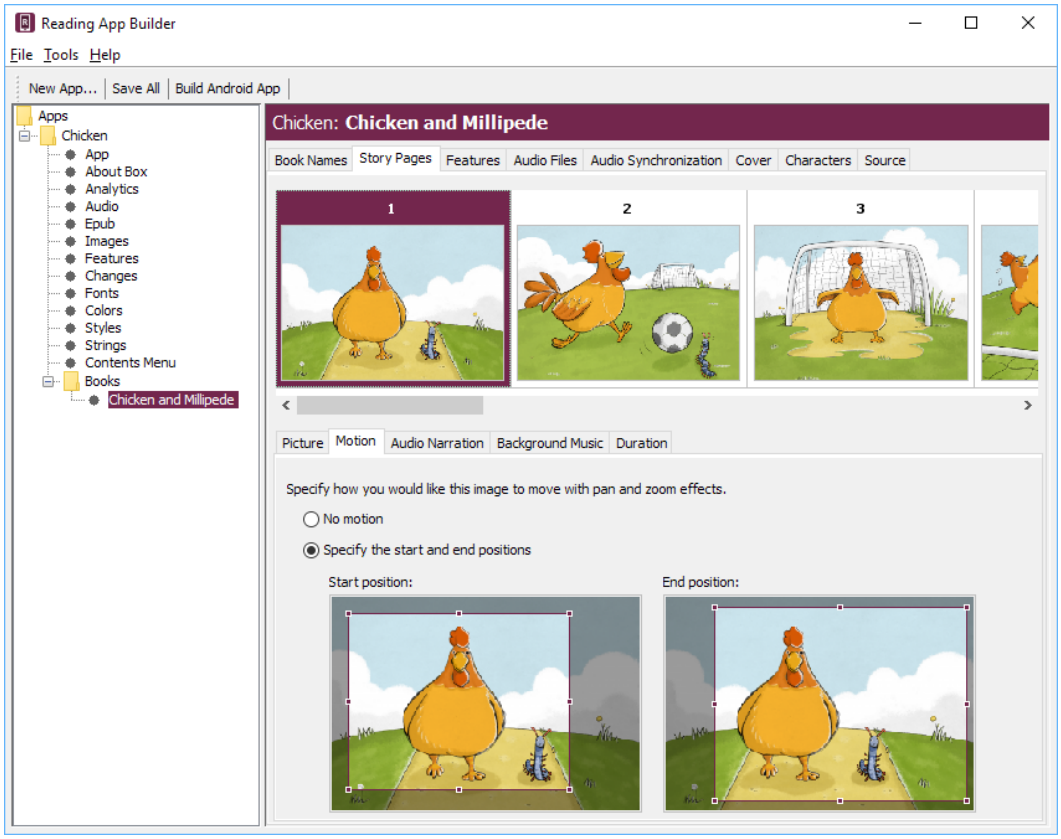
If you have defined the picture story book as an SFM text file, you need to add the images to Reading App Builder separately. Do this on the **Media** ➢ **Illustrations** page. You can use either PNG or JPG files. Try and make them as small as possible without compromising image quality. This will keep your app size small and reduce page load time.

[1] *Chicken and Millipede*, Copyright © 2014, African Storybook Initiative. Creative Commons.

12.5. How can I get the pictures to move when the audio is playing?

You can add pan and zoom effects to the images when the audio is playing. To specify the motion:

1. Go to the **Story Pages** tab for the book.
2. Select an image at the top of the page.
3. Select the **Motion** tab below the images.
4. Select **Specify the start and end positions**.



5. Drag and resize the start and end rectangles in the images to specify the start and end positions.
6. Do this for each image in the book.

12.6. What about font and font size?

A popular font designed for those learning to read is **Andika** or **Andika Compact**.

If you have a small amount of text per picture, it is best to use a larger font size by default. You can change this within Reading App Builder on the **Appearance** ➢ **Styles** ➢ **Text Styles** page by modifying the **font-size** property of the **body** style.

12.7. What audio timing labels are used?

If you have one audio file for the whole book, timing labels are of the form:

1a - 1st phrase on page 1
1b - 2nd phrase on page 1
1c - 3rd phrase on page 1
2a - 1st phrase on page 2
2b - 2nd phrase on page 2, etc.

If you have one audio file per page, timing labels are of the form:

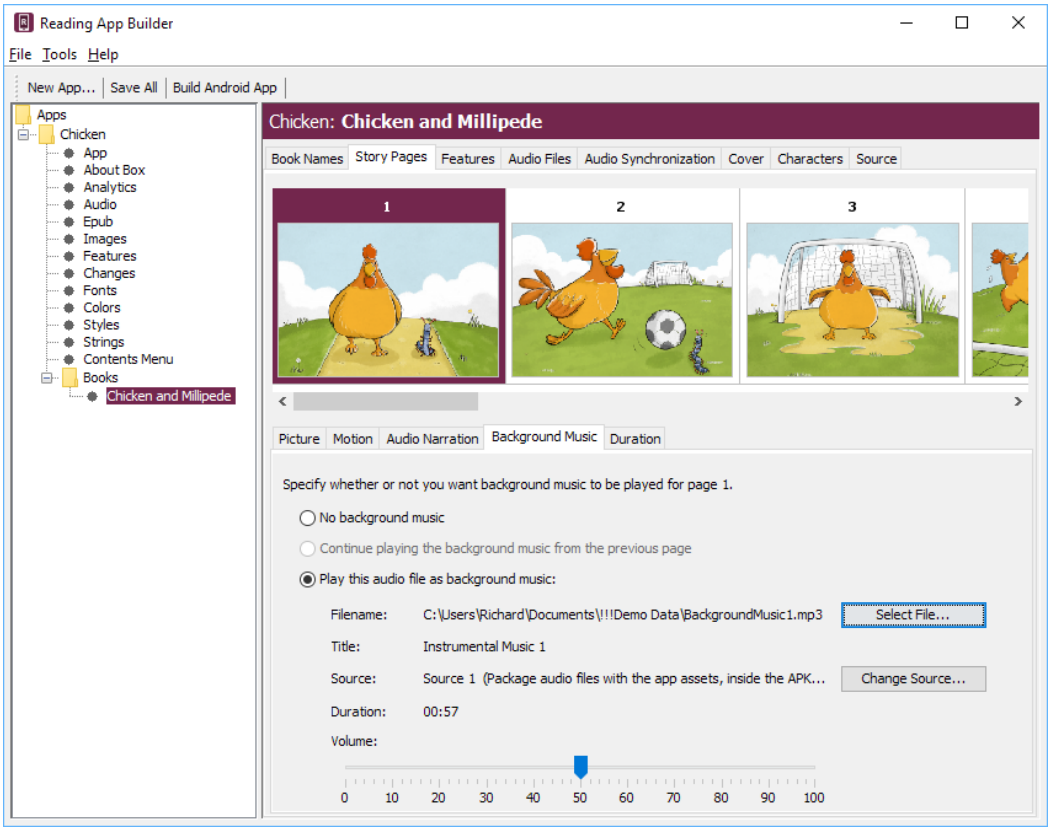
a - 1st phrase on the page
b - 2nd phrase on the page
c - 3rd phrase on the page

To help with the creation of the timing files, generate phrase lists on the **Audio Synchronization** page for the book, or use **aeneas** to automate the synchronization process.

12.8. How can I add background music when the audio is playing?

You can add background music and sound effects behind the audio narration track. To do this:

1. Go to the **Story Pages** tab for the book.
2. Select the first image at the top of the page.
3. Select the **Background Music** tab below the images.
4. Select **Play this audio file as background music**.
5. Click **Select File…** and choose the background music audio file.



6. Drag the slider to modify the volume of the background music.

7. Select each of the following pages in the book and specify whether to Continue playing the background music from the previous page, whether to play a different background music track or whether to stop playing any background music.

12.9. How can I record the audio files?

Use your favourite sound recording software, such as Audacity. You'll need to decide how fast or slow you want the reader to read. In some contexts, it might be good for picture story books to be read fairly slowly.

13. Song Books

13.1. How do I define a song book?

You can build a song book app, with one song per 'chapter'. Song books are defined in one of two formats:

- Microsoft Word (.docx) documents, or
- Text files using standard format markers (SFM).

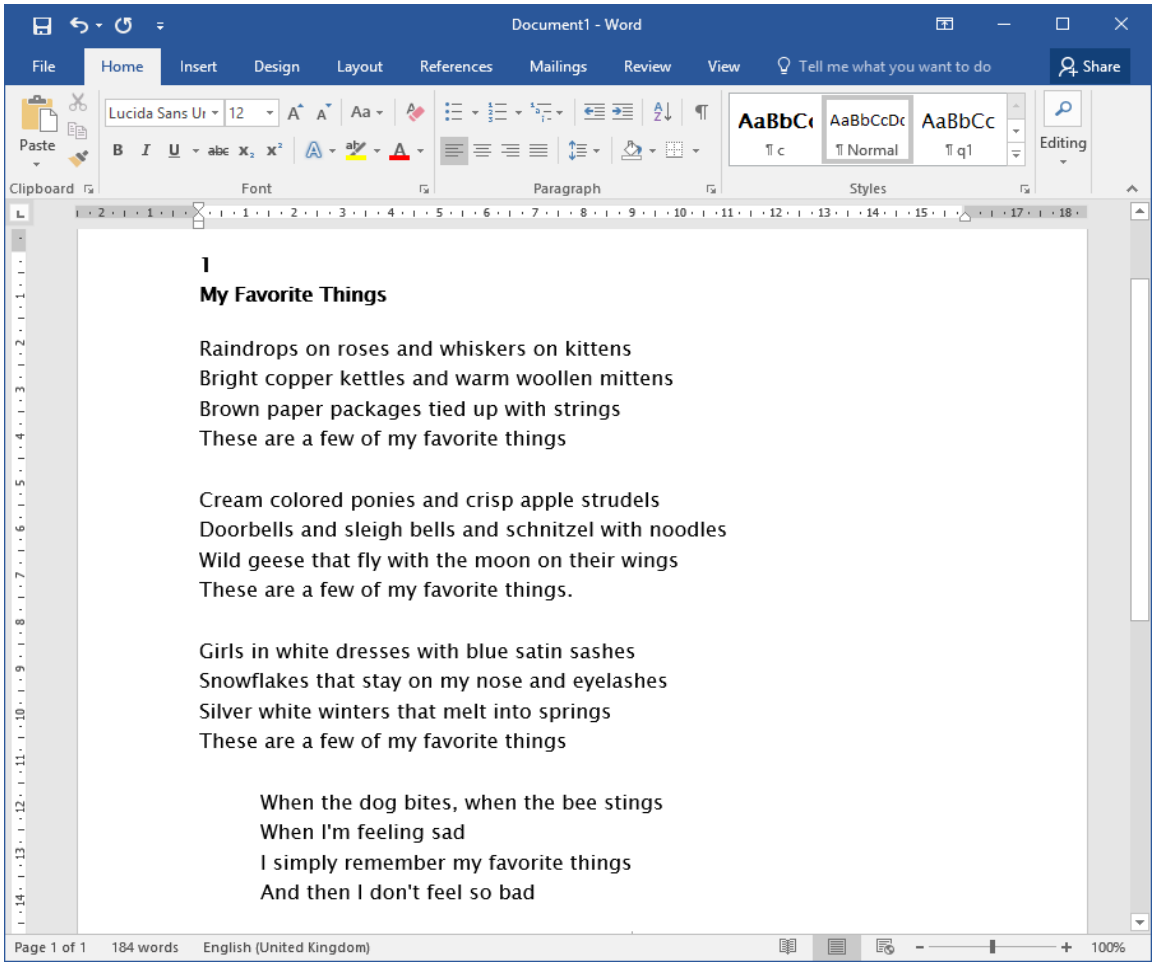
See 12.2 and 12.3 for a description of these formats.

Audio can be added with audio-text synchronization so that each line of the song is highlighted as it is sung.

13.2. What do song books in Word documents look like?

To create a Microsoft Word document to define a song book:

1. Start with a new blank document in Word.
2. Type the songs or copy them from another document.
3. For each song, place the song number on a line by itself before its corresponding title and lyrics.



4. Create the following paragraph styles:

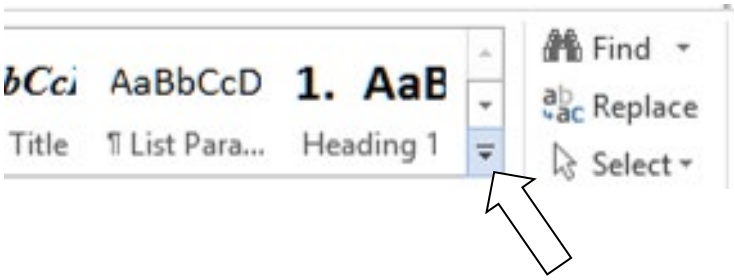
Style	Use
c	Chapter number

Style	Use
s	Song title
q1	Line of the song
q2	Line of a chorus (more indentation)

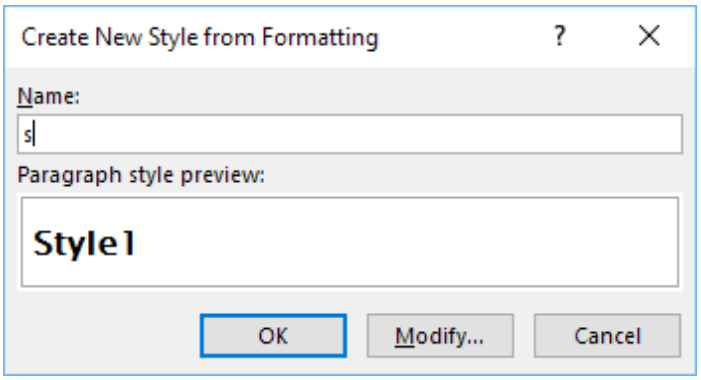
To create a style:

a. Select the line/paragraph of text that corresponds to this style, i.e. if you are creating the style 'c', select one of the lines with a song number. If you are creating the style 'q1', select a line of the song lyrics.

b. Click on the down arrow to the right of the styles ribbon.



c. Then select **Create a style** from the menu that appears.



d. Type the style name into the dialog box and press OK.

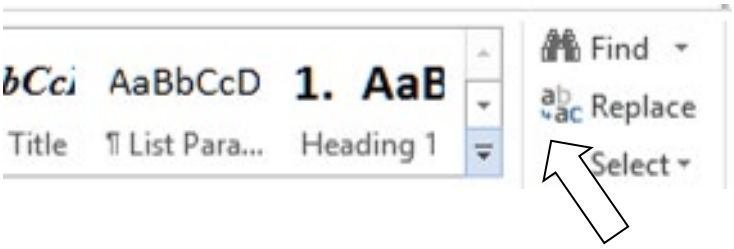
Do this for c, s, q1 and q2.

5. Apply the above styles to all the chapter numbers, song titles and the lines of the song lyrics.

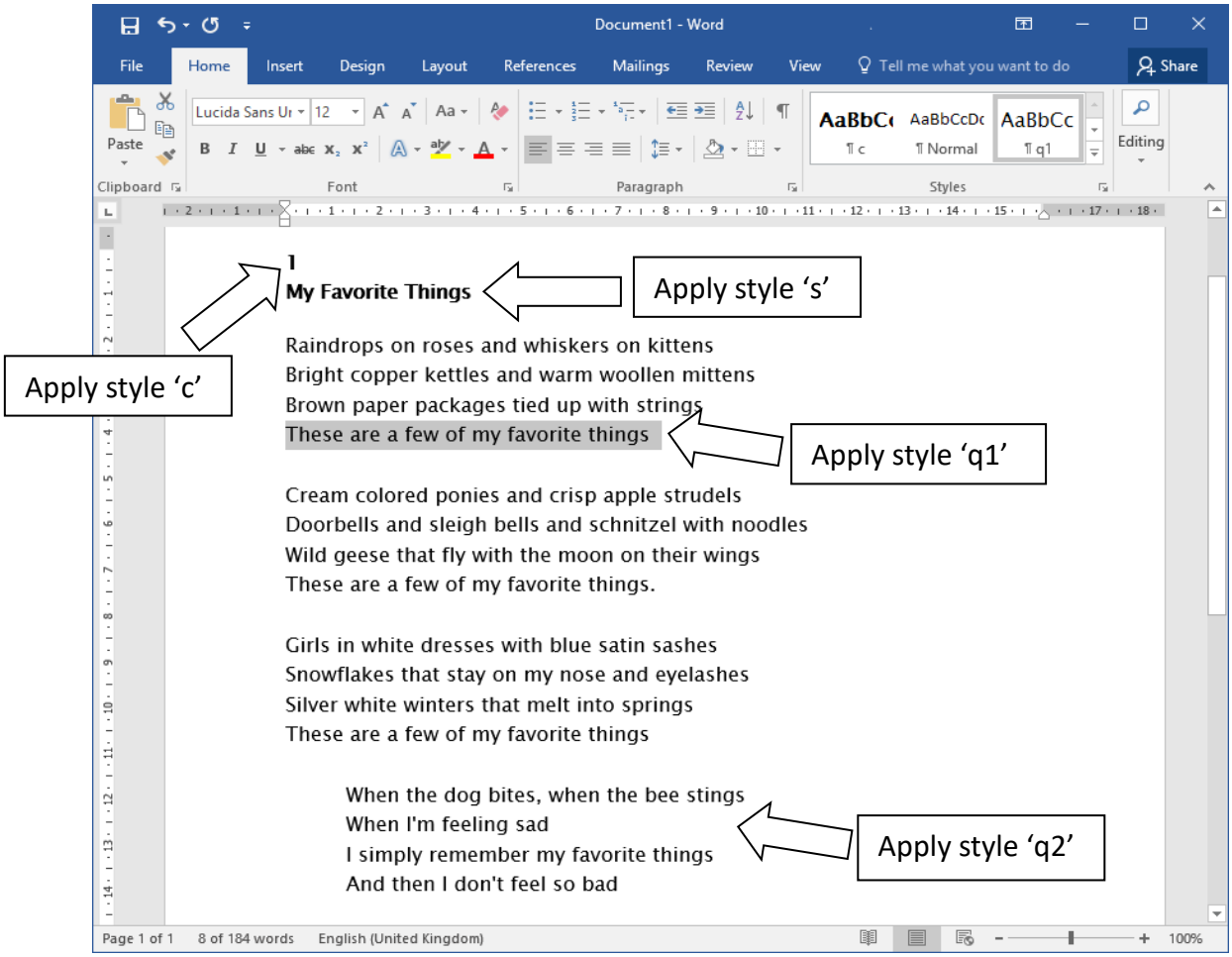
To apply a style:

a. Select the line/paragraph to which you want to apply the style.

b. Click on the down arrow to the right of the styles ribbon.



c. Select the style you want to apply.



Tip: The F4 key can be useful as you apply the styles to repeat your previous action.

All of the songs are contained within a single Word document.

6. Save the document in **.docx** format.

13.3. What do song books in SFM format text files look like?

They start with an id of your choice:

```
\id SONGS
```

and a title:

```
\toc2 Our Song Book
```

Each song begins with the **\c** marker and the number of the song:

```
\c 1
```

The marker **\q** (or **\q1**) is used for line of the song lyrics and **\q2** for indented lines, such as for a chorus.

Use **\b** for blank lines between verses.

```
\id SONGS
\toc2 Song Book

\c 1
\s My Favorite Things

\q1 Raindrops on roses and whiskers on kittens
\q1 Bright copper kettles and warm woollen mittens
\q1 Brown paper packages tied up with strings
\q1 These are a few of my favorite things
\b
\q1 Cream colored ponies and crisp apple strudels
\q1 Doorbells and sleigh bells and schnitzel with noodles
\q1 Wild geese that fly with the moon on their wings
\q1 These are a few of my favorite things.
\b
\q1 Girls in white dresses with blue satin sashes
\q1 Snowflakes that stay on my nose and eyelashes
\q1 Silver white winters that melt into springs
\q1 These are a few of my favorite things
\b
\q2 When the dog bites, when the bee stings
\q2 When I'm feeling sad
\q2 I simply remember my favorite things
\q2 And then I don't feel so bad
\b
\q1 Raindrops on roses and whiskers on kittens
\q1 Bright copper kettles and warm woolen mittens
\q1 Brown paper packages tied up with strings
\q1 These are a few of my favorite things
```

```
\c 2
\s Edelweiss

\q1 Edelweiss, edelweiss
\q1 Every morning you greet me
\q1 Small and white
\q1 Clean and bright
\q1 You look happy to meet me
\b
\q1 Blossom of snow
\q1 May you bloom and grow
\q1 Bloom and grow forever
\q1 Edelweiss, edelweiss
\q1 Bless my homeland forever
```

If you are used to using Paratext, these SFM codes will be familiar since they are how you mark up poetry.

All of the songs are contained within a single file.

The SFM songbook file must be a **plain text file**. If you have Unicode characters, the text files should use **UTF-8 encoding**. To create a text file in Windows, use Notepad. To create a text file on a Mac, use TextEdit, remembering to choose Plain text files in the preferences because otherwise the default file type is RTF (which contains a lot of additional formatting codes).

13.4. How can we associate audio with each song?

A separate audio file and corresponding timing file needs to be created for each song.

Timing labels are of the form:

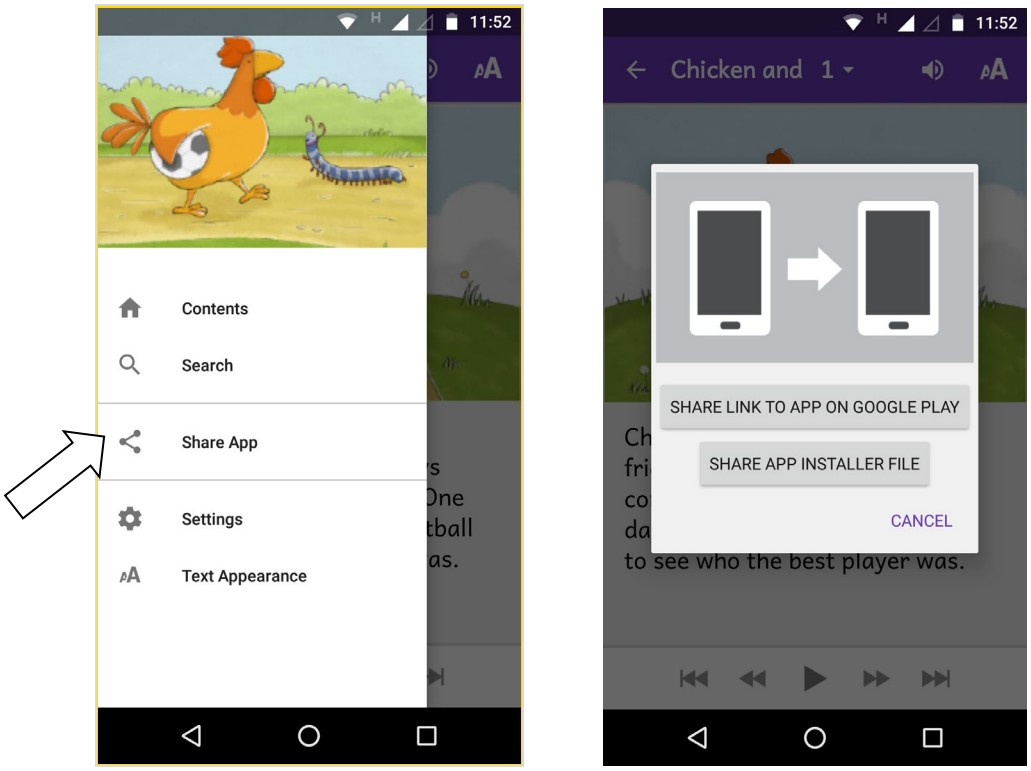
a - 1st line of the song
b - 2nd line of the song
c - 3rd line of the song
d - 4th line of the song, etc.

To help with the creation of the timing files, generate phrase lists on the **Audio Synchronization** page for the book.

Automated synchronization is unlikely to work for songs, so you will need to use the manual tagging method using Audacity and Timing Labels Assistant. Please see the document *Using Audacity for Audio Text Synchronization* for instructions.

14. Sharing Apps

A **Share App** item can be added to the app navigation drawer to enable users to share the app with others. You can configure this on the **Navigation** ➢ **Navigation Drawer** ➢ **Sharing** page.



There are two ways of sharing the app. You can choose to allow the user to use one or both of them:

- **Share Link to App on Google Play**: this option allows the user to send someone the link to the app on the Google Play store, e.g. by email or social media. You should enable this only if the app has been published on Google Play.
- **Share App Installer File:** this option allows the user to share the APK file with another device, e.g. by Bluetooth, Wifi transfer or email. This option is especially useful in contexts where not everyone has easy internet access and where you want to promote offline app sharing.

15. Deep Linking

Deep Linking allows you to direct people to specific pages within the app, using a link on a website, in a messaging app, social media or email.

A deep link can look something like this:

tharaka-stories://B002

or

https://tharaka.org/stories/B002

or

https://www.example.com/tharaka/stories?ref=B002

If you share one of these links with a friend on WhatsApp, Facebook or email, and they tap on the link on their smartphone, the app (if already installed) will open and go directly to story book 2.

15.1. Setting up Deep Linking

To set up deep linking in your app:

1. Go to the **App** ➢ **Deep Linking** page within Reading App Builder.
2. Select **Enable Deep Linking**.
3. Specify one or two link **schemes** to be associated with your app: a custom URI and/or a URL.

Custom URI Scheme

The first type of link scheme is a **custom URI** (Uniform Resource Identifier). This is a sequence of characters beginning with a letter and then any combination of letters, digits, plus (+), period (.), or hyphen (-). It is followed by a colon and two forward slashes (://).

Here are some examples:

mamara-stories://
booksmalinke://
proverbstoday://
tharaka-health://
literacy-tanzania://

Make sure that you do not choose a custom URI already in use by a popular app, such as:

fb://	Facebook
twitter://	Twitter
whatsapp://	WhatsApp

youtube:// YouTube

URL Scheme

The second type of scheme is a **URL** (Uniform Resource Locator), which is like a web address. It begins with http:// or https://. It can include a path.

Here are some examples:

https://mamara.org/
https://mamara.org/stories/
http://www.literacy-tanzania.org/
https://wycliffe.org/mexico/huichol/stories/

15.2. Creating Deep Links

This section explains how you create deep links to content in the app. We will look at how to specify book references (books, chapters), and how to turn the audio toolbar on when the app opens.

Specifying book references in a deep link

A reference is specified in the form BBB.P, where BBB is the book ID and P is an optional page number.

Examples:

B002.15	Book 2, page 15
B004	Book 4
B010.1	Book 10, page 1

Please note that the reference is case insensitive, so b001.15 and B001.15 are recognised as referring to Book 1, page 15.

There are two ways of specifying the reference in a deep link:

Method 1: Append the reference immediately after the link scheme, e.g.

mamara-stories://B002.3
https://mamara.org/stories/B002.3

Method 2: Specify the reference in a query string, after '?' and 'ref=', e.g.

mamara-stories://?ref=B002.3
https://mamara.org/stories?ref=B002.3

When someone taps on this link, the app (if installed) will open at the specified book and page.

Turning on the audio toolbar in a deep link
You can turn on the audio toolbar when the app launches, with the addition of the **audio=1** parameter in the query string of the deep link.

Here are some examples:

mamara-stories://B002?audio=1
https://mamara.org/stories/B003.2?audio=1

Note that a query string begins with the question mark (?) character before the first parameters, and then an ampersand (&) before subsequent parameters. So if you are using the 'ref=' method of specifying a reference, you will need to use '&', e.g.

mamara-stories://?ref=B002&audio=1
https://mamara.org/stories?ref=B003.2&audio=1

15.3. Deferred Deep Linking

The deep links described above will only work if the user already has the app installed. If they do not have the app, when they tap on the link, their device will not recognise it and an error message will appear.

Branch (branch.io) provides a way of handling **deferred deep links**. If a user taps on a link to the app but does not have the app installed yet, the link will first redirect the user to the app store to download the app. Then after the app is installed and launched for the first time, the user will be taken to the deep link page.

Setting up the app to use Branch
To use Branch for deferred links:

1. Go to the **Branch** website, https://branch.io/ and **Sign Up** for an account if you do not already have one.

2. Create a new app in the Branch dashboard.

3. Go to the **Link Settings** page, find the **Android** section and click the checkbox 'I have an Android App'.

4. Enter the custom URI scheme and where to download the app (e.g. on Google Play).

5. Within Reading App Builder, go to the **App** > **Deep Linking** page.

6. Select **Use Branch for Deferred links**.

7. Enter the **Branch Key** for this app. This can be found on the **Account Settings** page in Branch and will be something like:

   key_live_abCdEF2GQ7ID9rLPlxk91khczmpGZl71

8. Enter the **Default Link Domain** for this app. This can be found on the **Link Settings** page in Branch and will be something like:

   mamara-stories.app.link

Creating deferred links
Deep links using Branch will look something like this:

https://mamara-stories.app.link?ref=B002

where mamara-stories.app.link is the **Default Link Domain** specified on the Branch **Link Settings** page.

Please note that you need to use the **?ref=** method of specifying the reference. You cannot use https://mamara-stories.app.link/B002.

16. Analytics

If you enable Analytics, the app will connect to the internet from time to time to send app usage information to one or more analytics accounts. This will give you an idea of the extent to which people are interacting with the app.

The information sent will include the model of the device (such as 'Google Nexus 7', 'Samsung Galaxy S4'), the Android version (such as '4.2'), the mobile network provider and an approximate location (city/country). No personal information is included.

You can configure your app to send usage data to one or more of the following analytics engines:

Firebase Analytics	Sends data to a Google Firebase Analytics account of your choice.
Amplitude Analytics	Sends data to an Amplitude account of your choice.
S3 Digest Analytics	Sends a digest of analytics data to an Amazon S3 Bucket of your choice.

To set up analytics:

1. Go to the **Data & Analytics** ➢ **Analytics** page for the app.
2. Select **Enable Analytics**.
3. Click **Add Analytics Account…**
4. Choose an account type and enter your analytics account information.
    - For Firebase Analytics, you will need a google-services.json configuration file for your account.
    - For Amplitude Analytics, you will need an API Key
    - For S3 Digest Analytics, you will need an S3 Bucket ID and an Identity Pool ID.

16.1. Firebase Analytics

To sign up for Firebase Analytics, ensure you have a Google account.

You will need to:

1. Go to the Google Firebase website at https://firebase.google.com/ and ensure you are signed in with your Google account.
2. Click **Add project** to create a Firebase project for this app.
3. In Step 1, you will need to give your new project a **Project name**.
4. In Step 2, titled **Google Analytics for your Firebase project**, select **Set up Google Analytics for my project** and press **Continue**.
5. In Step 3, titled **Configure Google Analytics**, click on the drop-down box and choose **Create a new account**. Give it a name, which can be the same as your Firebase project name.
6. Check the boxes to confirm that you accept the analytics and data protection terms. Click **Create project** and wait a few seconds for the project to be created.
7. Click the Settings button (a cog wheel icon near the top) and select **Project settings**.

8. On the **General** tab of the Settings, scroll down to the **My apps** section and click the Android app icon.
9. On the **Add Firebase to your Android app** page, enter your app package name and click **Register app**.
10. Download the config file, **google-services.json**. That is all you need from the app registration. You can ignore the information on the rest of the screens and return to Settings.

11. Go to the **Data & Analytics** ➢ **Firebase** page in Reading App Builder.
12. Select **Firebase Analytics** as one of the features to use in the app.
13. On the **Firebase Configuration** ➢ **Android** tab, click the **Browse** button and find the **google-services.json** config file that you have just downloaded from the Firebase console.

16.2. Amplitude Analytics

To use Amplitude analytics, you will need to create an account. Go to:

https://amplitude.com

You will need to:

1. Click **Sign Up** at the top right of the screen and create an account. You will receive an email to finish activating your account. Copy the link to your browser and go to the page and complete the activation process.
2. You will be prompted to create a new organization. You can do that to invite team members to join your organization and access the data. You will also be prompted for some additional information.
3. Click **Create Project**, enter the project name and click **Create**. There will be a project for each individual app.
4. Click **Projects** on the left of the screen. This will show the list of projects and the properties including a long string of hexadecimal characters under the label **API Key**.
5. Highlight and copy this string into the API Key field in Reading App Builder.

16.3. S3 Digest Analytics

To use S3 Digest Analytics, ensure you have admin permissions to an Amazon AWS account, and go to:

https://aws.amazon.com/console/

You will need to:

1. Click **Sign In to the Console** at the top right of the screen.
2. Create an S3 Bucket.
    a. Go to the **S3** Service and click **Create bucket**, enter a Bucket name, select a Region near where the app will be distributed, and click **Next**.
    b. On the **Set properties** and **Set permissions** steps, use the defaults and click **Next**.
    c. Review the configuration and click **Create bucket**.
    d. Copy the **Bucket name** into the **S3 Bucket ID** field in Reading App Builder.

3. Create a Federated Identity.
    a. Go to the **Cognito** Service and click **Manage Federated Identitites**. The first time you use this service it will start creating an identity pool for you. If the AWS account already has identity pools, then it will show a grid of existing one. If this is the case then click **Create new identity pool**.
    b. Enter an **Identity pool name**, click **Enable access to unauthenticated identities** (which allows users of the app to submit analytics without logging into some service), and click **Create**.
    c. Click **Show Details** to see the **Role Name** for the unauthenticated identities (in the next step, we will give them permission to put objects in the bucket created in the previous step). Click **Allow**.
    d. Copy the **Identity pool ID** value inside the quotes in the **Get AWS Credentials** section of the **Sample code** page shown after completion of the previous step. Copy this string into the **Identity Pool ID** field in Reading App Builder.
4. Give permission to put data into the S3 Bucket.
    a. Go to the **IAM** Service and click **Roles** category.
    b. Click on the Role created in step #3 (e.g. Cognito_<IdentityPoolName>Unauth_Role).
    c. Click **Add inline policy**, click **Service** and choose **S3**.
    d. Click **Actions**, type in *PutObject* to search for actions, and check **PutObject** to select that action.
    e. Click **Resources**, use default Specific, click on **Add ARN** link, enter the **Bucket name** from step #1 into the **Bucket name** field, click the **Any** checkbox at the end of the **Object name** field, and click **Add**.
    f. Click **Review policy**, enter a name in the **Name** field, and click **Create policy**.

16.4. Data Payload for S3 Digest Analytics

S3 Digest Analytics will send a small daily digest (about 300 bytes for each day the app is used) of compressed, completely anonymous data (no personal, phone, or GPS location information) via an encrypted transport (https) to an Amazon data center, when the device is connected to the Internet (via cell or WIFI) while the app is running. You are responsible for hosting the Amazon S3 Bucket and processing the received data.

Files are uploaded to the S3 Bucket and in a sub-folder based on the format (e.g. the current format is f1) and stored with a unique filename using a randomly generated GUID as the basename. We are working to package a Splunk configuration so that you can deploy your own server to analyze the data.

S3 Digest Analytics uses a JSON payload format. A complete sample data payload is below:

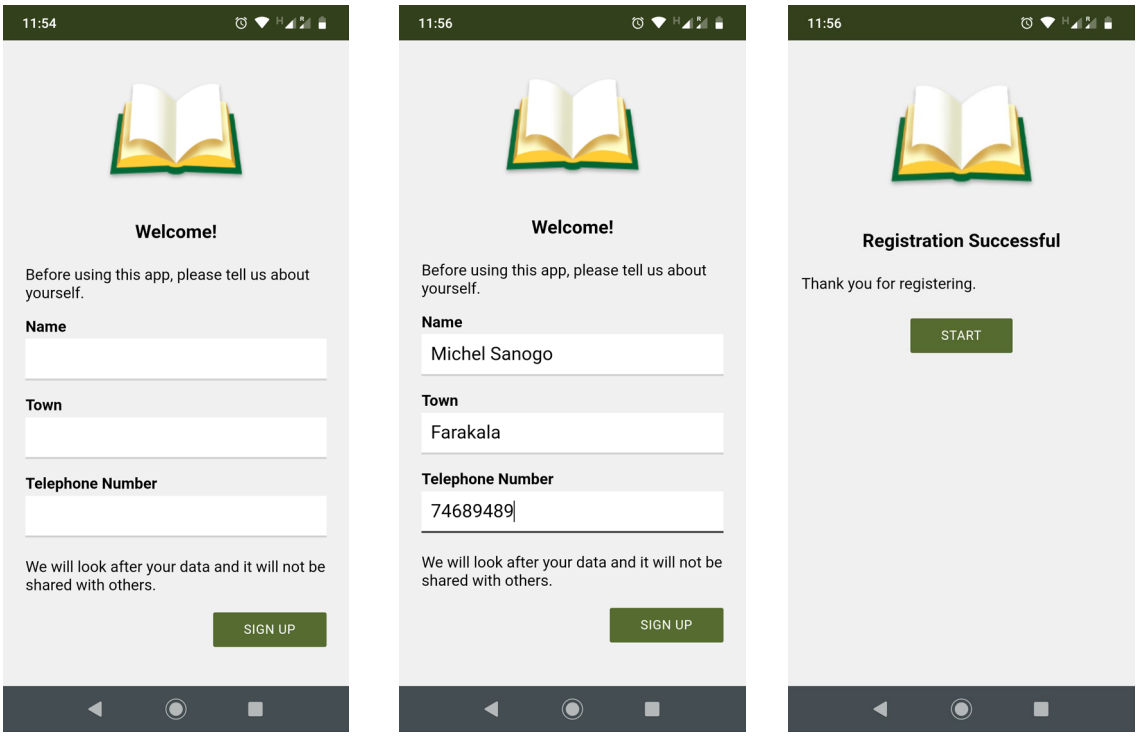
```
{"startTime":"20180306T0835Z","period":1440,"id":"12db7e3f-93d9-4370-b12b-
fe048804e4f5","package":"org.sil.picturebooks.cuk","version_name":"1.0.1",
"sessions":1,"sessionMins":21,"shares":3}
```


This sample is comprised of the following fields:

- One day of activity (1440 minutes), starting on 2018-03-06
- The id is a GUID which was randomly generated on the phone when the app was first launched, enabling determination of how many unique installations of the app are in use (but no user-identifying information).
- Package and version indicate which app is in use.
- This report was for a single 21-minute session. This (and other) values would be incremented if the app had been used multiple times within the reporting period.
- The user pressed share in the app 3 times.

17. Registration Screen

You can define a registration screen to be shown when a user launches the app for the first time. This allows you to collect contact information, for example to connect people with reading groups.



To set up a registration screen, you need to do some configuration work in two places:

- within Reading App Builder, and

- in the Google Firebase console.

17.1. Setting up the Registration Screen in Reading App Builder

To set up the registration screen within the app builder:

1. Go to the **App ➢ Security** page.
2. Select **Require each user to register with their details when they first use the app**.
3. Click the **Configure Registration** button.
4. Follow the instructions in each of the tabs in the Configure Registration dialog:
    i. **Registration Screen:** specify the title and text to show on the registration screen.
    ii. **User Details:** specify the input fields (such as name, telephone number, email adress), that you will ask the user to provide. You can specify which of these are required and which are optional.
    iii. **Skip Registration:** choose whether you want the user to be able to skip the registration process. If they do press the Skip button, specify when you want the app to ask them again.
    iv. **Registration Completed:** specify the title and text to be displayed on the screen which is shown to the user after they have successfully registered.
    v. **Image:** specify an image to be displayed at the top of the screen, such as your organisation's logo or the app icon.
    vi. **Database:** choose whether the user data will be stored in an app-specific path in the database (which is useful if you use the same database for several apps), or at the top level of the database (which is fine if you have just one app). More information about configuring your database can be found in the following section.
    vii. **Styles:** you can adjust the styling of the screens by adjusting the style declarations.

    Use the **Test** buttons on the **Registration Screen** and **Registration Completed** tabs to preview the screens in a browser.

5. Click **OK** when you are finished in the dialog.

17.2. Setting up the database in the Google Firebase console

To set up the database, which will contain the registered users' information, you need to add Firebase to your app, create a database, set up authentication and configure rules.

Add Firebase to your app

To add Firebase to your app:

1. Go to the Google Firebase website at https://firebase.google.com/ and ensure you are signed in with your Google account.
2. Create a Firebase project if you do not already have one for this app.
3. Click the Settings button (a cog wheel icon near the top) and select **Project settings**.

4. On the **General** tab of the Settings, scroll down to the **My apps** section and click the Android app icon.
5. On the **Add Firebase to your Android app** page, enter your app package name and click **Register app**.
6. Download the config file, **google-services.json**. That is all you need from the app registration. You can ignore the information on the rest of the screens and return to Settings.
7. Go to the **Data & Analytics** ➢ **Firebase** page in Reading App Builder.
8. Select **Firebase Realtime Database** as one of the features to use in the app.
9. On the **Firebase Configuration** ➢ **Android** tab, click the **Browse** button and find the **google-services.json** config file that you have just downloaded from the Firebase console.

Create a Database

To create a **Realtime database**:

1. In your Firebase project console, select **Database** from the menu on the left of the screen.
2. Scroll down the screen for the section titled **Or choose Realtime Database** and click the **Create database** button.
3. Choose **Start in locked mode** as the Security rules and click **Enable**.

Set up Authentication

On the **Authentication** page of the Firebase console, enable two authentication types on the **Sign-in method** tab:

1. **Email/Password** (this is used when viewing registered users in a web browser)
2. **Anonymous** (this is used during user registration in the app)

Configure Rules

Rules are required to tell Firebase who will have access rights to read and write to the database. To configure the database rules:

1. In your Firebase project console, select **Realtime Database** from the menu on the left of the screen.
2. Select the **Rules** tab.
3. The rules you specify will depend on the path you have chosen on the **Database** tab in the Registration configuration.

   If you have chosen to store the user information in an **app-specific location**, e.g. "/apps/your-app-package-name/users/", the rules need to be as follows:

```
{
  "rules": {
    "apps": {
      "$package": {
        "users": {
          "$uid": {
            ".read": false,
            ".write": "$uid === auth.uid"
          }
        }
      }
    }
  }
}
```

   If you have chosen to store the user information at the **top level of the database**, e.g. "/users/", the rules need to be as follows:

```
{
  "rules": {
    "users": {
      "$uid": {
        ".read": false,
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

   These rules give write access only to the user who is making the registration. No one will have read access (except for you when you view the database from the Firebase console).

Tip: Make sure you copy the rules exactly. An easy way to get the rules you need is to click the **Database Rules** button on the Database tab. This will give you the rules you can copy.

4. Click **Publish** to confirm your changes.

Rules FAQ

Q. If we want an administrator to have read access to the Registered users to display in a web browser, what rule do we use?

If you want to give an administrator read access to the data, to be displayed in a web browser (See **Configure Registration** ➢ **User Details** ➢ **View Data**), here is the rule to use:

```
{
  "rules": {
    "apps": {
      "$package": {
        "users": {
          "$uid": {
            ".read": "(auth != null) && (auth.token.email
== 'me@abcd.com')",
            ".write": "$uid === auth.uid"
          }
        }
      }
    }
  }
}
```

(where me@abcd.com is a user added on the **Authentication** page)

The above rules assume that you want to store the user information in an app-specific location. If you have chosen the top-level location, you need to remove the “apps” and “$package” elements above.

18. EPUB

As well as building a smartphone app, you can create e-books in EPUB format. EPUB documents are readable by a number of e-book readers on Windows, Mac, iOS and Android platforms.

If you have timing files for your audio, the e-books can contain ‘Read Aloud’ text-audio synchronization. These are readable by a few EPUB 3 readers, such as Adobe Digital Editions or Thorium Reader.

To create an EPUB e-book:

1. Choose whether to create a single e-book containing all the books in your collection or whether you want to create a separate e-book for each book. The latter is the best option if you will have audio in the book, since otherwise it could be a very large e-book.

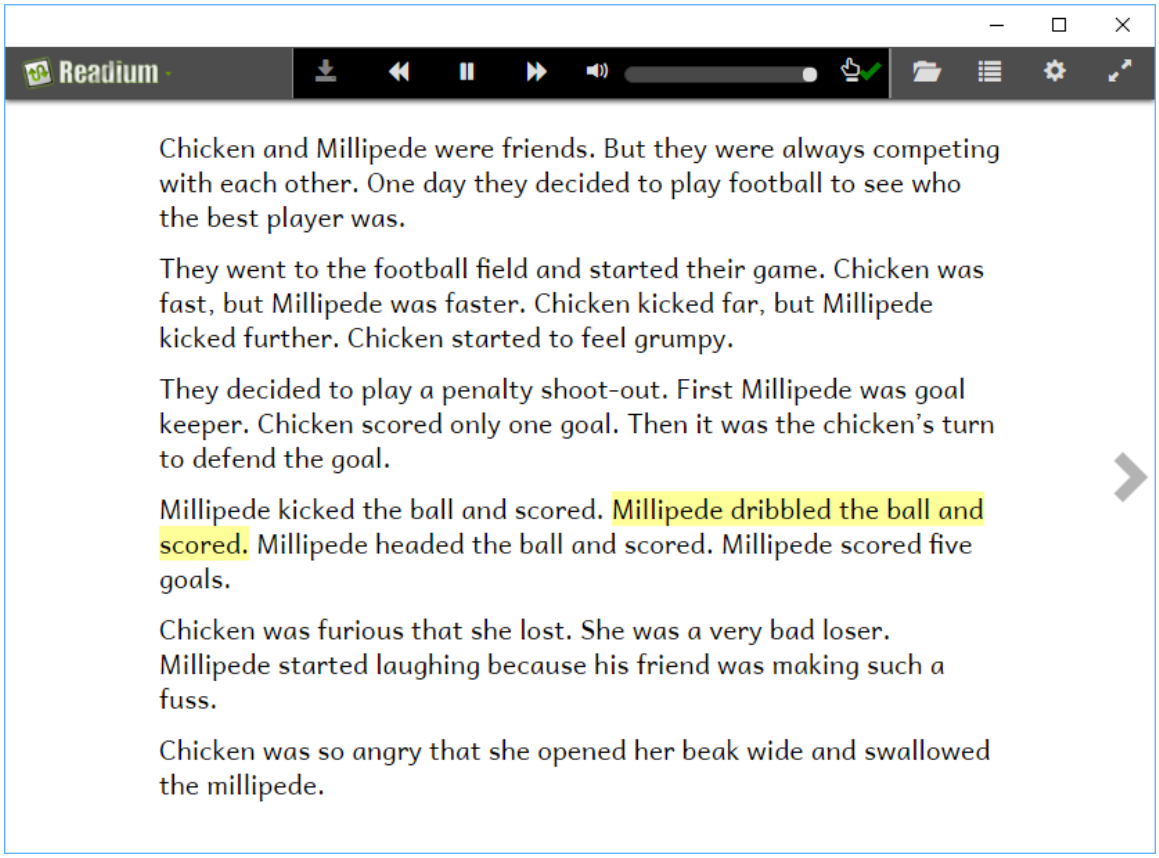
2. Choose a cover design. See the **Cover** tab. If you are creating separate e-books for each book, a different cover image can be specified for each book on the Cover tab of each individual Book page.

3. Go to **Publishing ➢ E-Books (EPUB)**, and click the button **Create EPUB Document(s)**.

   The generated EPUB documents will be saved in the default EPUB output folder. See **Tools ➢ Settings… ➢ Default Folders**.

4. Open the EPUB documents in your chosen e-book reader.

Reading App Builder generates EPUB 3 documents which should be backwardly compatible to be able to be read using EPUB 2 readers. EPUB 3-only features such as the text-audio synchronization will only be seen on compatible readers.



Viewing an EPUB Read Aloud document in the Readium Chrome extension

19. Publishing and Distribution

Before you distribute your app, please use the *App Publishing Checklist* (available via the Help menu in RAB or on the RAB website) to verify that it is ready to publish.

Apps built with Reading App Builder can be published on the Google Play store, distributed by memory card, shared by Bluetooth or Wi-Fi transfer, uploaded to websites, or sent out by email. For more information, please see the user manual *Distributing Apps* (available via the Help menu in RAB or on the RAB website).