Dolphin EasyReader for iOS and Android

**COMPANY CONFIDENTIAL**



# Platform requirements

* **Platform operating systems:**
  + Apple, two options (dependent on development cost, maintenance and test):
    - iOS 9 (at least 79% device coverage, 11 April 2016)
  + Android, two options (dependent on development cost, maintenance and test):
    - 4.4 KitKat (at least 74% device coverage, 11 April 2016)
* **Hardware devices**
  + iOS 9 gives us a list of iPhone 4S and forward, iPad 2 and forward.
  + Android gives as us more or less an unlimited(?!) set of hardware choices
* **Limitations**
  + We do not target Apple’s and Android’s TV and Wearables solutions at this point.
* **Screen sizes**
  + We need to support both Apple and Androids small and large screens (phone and tablet)

# General requirements

* **Overall design policy**The apps needs to follow Apple’s and Google’s general recommendations regarding design policy and behavior. However, by the nature of EasyReader, we may do some UI components a bit larger than default, use slightly different color schemes, icons, etc.
* **Screen rotation**  
  The apps needs to react on screen rotation and adapt the UI to the new screen layout and size. The apps need to make sure that UI components are positioned correctly when switching between landscape and portrait mode. This means that some UI components needs to be aligned e.g. top-left in a toolbar, playback buttons in the reader needs to be centered at the bottom and so on.  
  The apps do NOT need to introduce new UI elements due to landscape mode or larger screens.
* **Background processing**  
  The apps need to support some degree if background processing to mainly cover two needs:
  + **Ongoing playback –** the app should continue to playback the book when leaving the app. If some other app needs the audio controls, the app must stop the playback. The app doesn’t need to stop the playback on notification sounds (e.g. a “bing” for received email). It must also stop ongoing playback on incoming phone call.
  + **Book download** – we need to allow background downloads of books.
* **Locked screen**The apps need to display a limited set of playback controls on the device’s lock screen (similar to Spotify). Play/stop, navigate next/prev. However this should only be visible when a book is loaded in the reader. If the UI is somewhere else in the app (e.g. in the “My Books” area), the app shouldn’t display anything on the locked screen. For more info, see section [Locked screen playback controls](#_Locked_screen_playback)
* **Accessibility**The apps needs to work well with VoiceOver and TalkBack. We also have UI requirements for working well with people with dyslexia and low vision, but this is covered in the design prototypes and further below in this specification.
* **Application languages**The apps needs to support multiple languages and the user can switch language in the application settings. The development team needs to make sure a translation work can be made at as low cost as possible without too much trouble integrating new languages (or corrections) into the source code.   
  *To be discussed: Ideally, language files should be stored on the Dolphin server and automatically downloaded by the apps on startup (if updates is found). This would make it easy for Dolphin to do text corrections and deploy new language without forcing a new update through the app stores.*
* **Online/offline**The app needs to nicely handle Internet drops and switches between online and offline.
* **Gestures**We don’t think we need to implement any gestures at this point, but here is a few ideas where this can be applicable (for discussion later):
  + Open/close the “main navigation” screen – In “My Books”, swipe from the edge of the screen to bring up the main navigation screen. Swipe it back to close it.
  + Two finger tap – In the reader, this could do play/stop
  + Swipe left/right – In the reader, this could navigate next/prev (or HTML page scroll)
* **Bidirectional UI**  
  *Investigation point*: We need to investigate how to support both LTR and RTL user interface on both iOS and Android. In phase one, we will only release in LTR languages, but we expect the apps to be translated into languages where RTL is needed. How much does the development tools and SDKs help us with this?

# Architecture and modules

The apps contains two major modules (bookshelf and reader) that are built on cross-platform technology from QT ([www.qt.io](http://www.qt.io)). Both modules do not have any UI component – they can be treated as strict C++ modules. Both modules have documentation and these are available in separate documents. The QT modules are today used on the Windows platform and we are now brining those into iOS and Android.

**Bookshelf module** – this is the module that holds all information about all online libraries, how to search and browse for new books, downloads, newspaper subscriptions, “my books” information, book meta data and much more. This module provides a single interface towards +18 different API library services.

**Reader module** – this is the module that has support for multiple book and newspaper file formats and it provides a unified interface for displaying content on screen, navigation, etc.

**Dolphin Cloud** – this is the API service that takes care of synchronizing user data on the cloud, so that the user can continue on another device and platform. See separate section for more information.

**Platform UI** – on each platform (iOS and Android), the UI screens and “glue” code between all these modules needs to be developed for each platform. This whole specification is very much about “Platform UI”, the screens and behavior.

**Analysis modules** – to understand the apps and its user more in depth, we need to add some automatic analysis modules into the apps. Below is a few suggestions:

* **Crash reporting**  
  Crashlytics (<https://fabric.io/kits/ios/crashlytics>)
* **User behavior/activity**  
  Answers (<https://fabric.io/kits/ios/answers>)

# Overall user stories

**User story:** As a user, I want the EasyReader UI to be familiar cross devices/platforms so that I can focus on reading my books instead of spending time on learning a new UI when switching device/platform.

**User story:** As a user with multiple devices, I want EasyReader to keep track on the books I’m reading so that I can continue reading when I get home and use another device.

**User story:** As a user, I want my bookmarks, notes, etc to be saved in the cloud so that I can continue my study work on the book when switching device/platform.

**User story:** As a blind user, I want the app to work well with Voice Over/Talk Back so that I can use the app in an efficient manner.

**User story:** As a user, I want the apps UI to adapt nicely when rotating the screen so that I can read my book in landscape/portrait mode with ease.

**User story:** As a user, I want to be able to download a book in the background so that I can do other tasks on the device while waiting for the download to be completed.

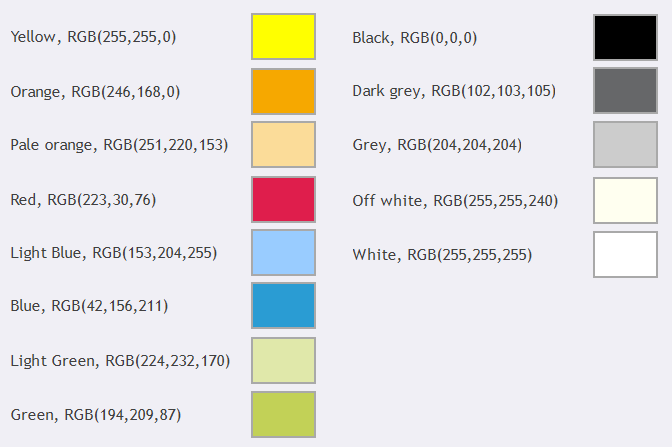
**User story:** As a user on the move, I want to have access to downloaded books in “offline mode” so that I can read my books when I’m at a location with bad Internet connection.

**User story:** As a user on the move (once connecting to Internet again), I want the app to automatically synch offline data so that I don’t lose any information when going online again. Alternative… so that I can trust my notes and bookmarks are saved securely in the cloud.

# Design

It is important that EasyReader on all platforms uses a common design and behavior theme, so that it is easy for a user to move between platforms and devices.

These are the colors of the new Dolphin branding:



The EasyReader apps will only use a few of the colors, with orange as its main accent color.

# Fonts

The reader has a font setting in the reader view where the user can apply a font to the book and EasyReader will bundle some “dyslexia friendly” fonts within the apps. These are the fonts we will ship with the apps:

* Comic Neue
* Libre Baskerville
* Open Sans
* OpenDyslexic
* Tiresias

# App screens

Below are all the screens needed in the apps

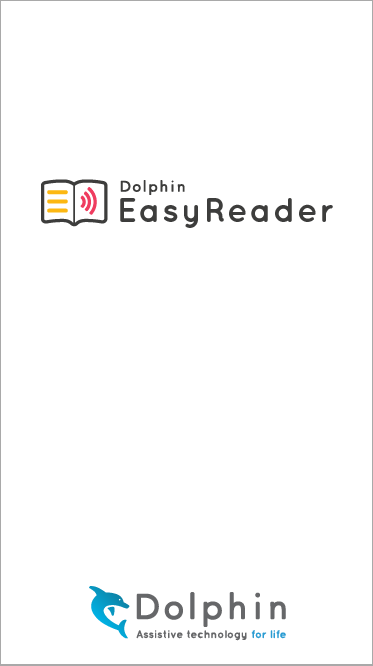
## First start up

When the app is loading, first it will show the splash screen for a certain amount of time (see separate section below). After the splash, the app will do one of the two:

* **Fresh install or not logged into EasyReader**  
  Display the [Getting Started screen](#_Screen:_Getting_started)
* **Logged into EasyReader**Display the [My Books screen](#_Screen_2:_My)

## Screen: Splash

When the app is loading, a splash screen will be shown for a short time. When this is shown, the app is initializing the bookshelf module and need to prepare the “My Books” screen. Dependent on Internet connection availability and speed, this duration may be different from time to time. When Internet connection is not available, the duration must be as short as possible. We need to implement a maximum duration for this splash and the bookshelf module has a default timeout for handling this.



The splash screen need to have this:

* the EasyReader logo, centered one third up on the screen
* the Dolphin logo, centered close to the bottom of the screen

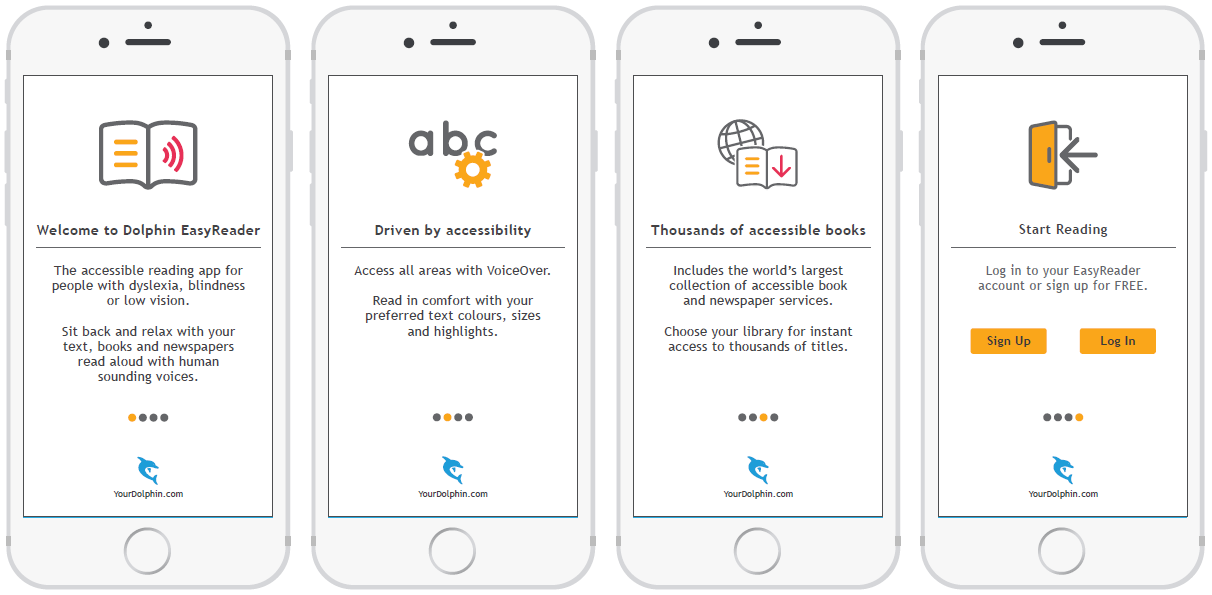
During first initialization of the app, it does this:

* Detect the device’s language code, which can be used for default values to bookshelf Init(), default libraries, default text prompt, default TTS voice, etc…
* Loads/init the bookshelf module, which connects to Internet and pulls down the list of available libraries.
* If there is a cached/saved login to EasyReader account, it should proceed to [My Books](#_Screen_2:_My) screen. If not logged in (or fresh install), it should now present the [Getting started](#_Screen:_Getting_started) screen.

NOTE: If no Internet is available, or the Dolphin server is down, a cached/saved login must be possible to be made so that the user can use EasyReader in offline mode. However, we need to discuss how long time it should be possible to use the app in offline mode without doing a proper Internet login. 3 days, 7 days, 1 month, no time limit?

## Screen: Getting started

This is a screen that contains a “slideshow” with some product information and the last screen have a signup/login feature similar to this:

Each slideshow screen contains at least this:

* Main graphics at the top
* Heading (larger font + bold)
* Horizontal line under the heading
* Some text (expect one or two paragraphs)

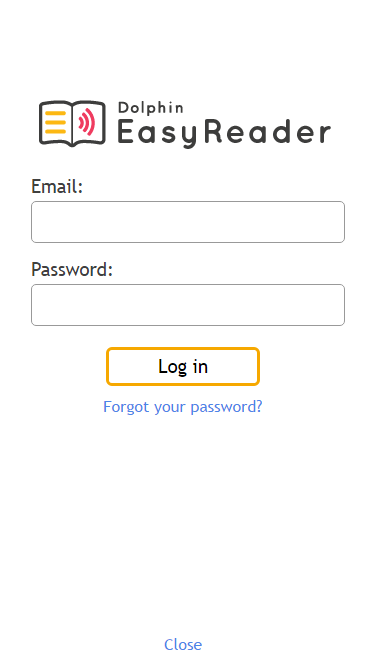
Under the slideshow screens are the slide page number indicator and a Dolphin logo. The Dolphin logo can be tapped on to launch external browser linking to this page: <https://yourdolphin.com/easyreader>

The last screen have a couple extra buttons:

* Sign up – that launches [the sign up screen](#_Screen:_Sign_up)
* Login – that launches [the login screen](#_Screen:_Login_1)

### Screen: Login

This is a “standard” login screen for logging into EasyReader.



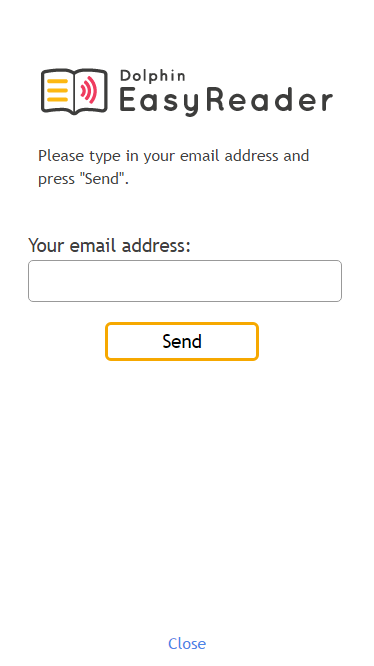
This screen contains:

* EasyReader logo
* Email label + edit area
* Password label + edit area
* “Login” button
* “Forgot your password?” link, that displays the [Forgot Password screen](#_Screen:_Forgot_password)
* “Close” link, that will take you to the [Getting Started screen](#_Screen:_Getting_started)

When pressing the “Login” button, the app will send the email and password to the Dolphin server for verification. If successful the app will save the login credentials in the app storage (to do auto login next time the app starts) and continue to the [My Books screen](#_Screen_2:_My).

### Screen: Forgot password

If the user have forgotten their password, this screen is shown.

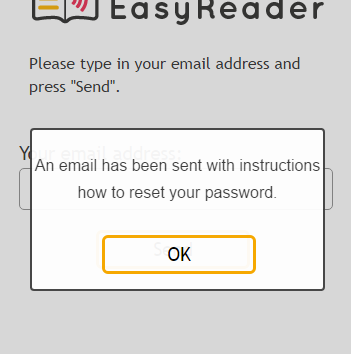


This screen contains:

* EasyReader logo
* Label with some text
* Email label + edit area
* “Send” button
* “Close” link, that will take you to the [Getting Started screen](#_Screen:_Getting_started)

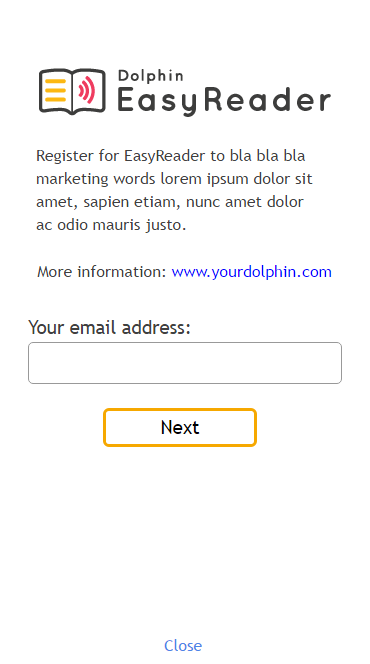
When pressing the send button, the app will send the email to the Dolphin server which will result it one of two messages:

* “An email has been sent with instructions on how to reset your password”  
  Pressing OK will take you back to [the Login screen](#_Screen:_Login_1).
* “That email address wasn’t found. Please try again.  
  Pressing OK will stay on the same screen with focus in the email edit area.



### Screen: Sign up

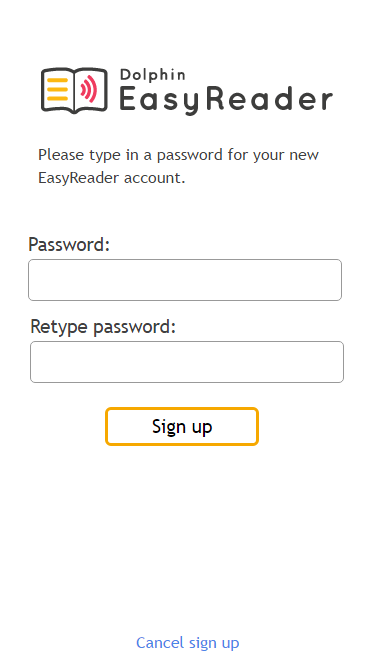
To be able to use the EasyReader app, the user need to acquire a free EasyReader account and this can be made from within the app itself.



This screen contains:

* EasyReader logo
* Text area with marketing words the benefits for signing up
* Possibly with a link to EasyReader’s web page for more information  
  <https://yourdolphin.com/easyreader>
* Label with the text “Your email address:”
* Text edit area, where the user types in the email address
* “Next” button – we need a simple verification of the text edit area so that the string looks like an email address. If it doesn’t meet the requirements, pop up a message “Please type in your email address.”  
  If the email address looks OK, the app goes to the next screen below (password).
* The small “Close” text/button/link takes you back to the [Getting Started screen](#_Screen:_Getting_started).

When pressing the “Next” button, the app will communicate with the Dolphin server and check if the email is free to use. If not free, a message will be displayed: “This email already have an EasyReader account.” If the email address is free to use, the app will now display the next screen for the account sign up:

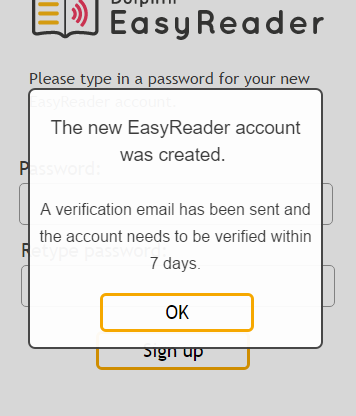


This screen contains:

* EasyReader logo
* Label with short text
* Label with the word “Password:”
* Text edit area (with password protection)
* Label with “Retype password:”
* Text edit area (with password protection)
* Button with the text “Sign up”
* A button/text/link that cancels this screen and takes you back to the [Getting Started screen](#_Screen:_Getting_started)

The user have to type in the password for the new EasyReader account two times and they need to match. This is validated when pressing the “Sign up” button. If the password isn’t matching, pop up a message “Passwords doesn’t match. Please try again.”

If the passwords are the same, the app will communicate with the Dolphin server to create a new EasyReader account. When this is done and confirmed by the server, this message is shown:



This means that the user have to verify the new EasyReader account within a certain time period. We need to discuss the suitable time for this verification.

When pressing OK, the app will display the [My Books screen](#_Screen_2:_My).

## Screen: My Books

**User story:** As a user, I want the app to list the books I’m currently reading so that I easily can find my book.

**User story:** As a user, I want easily open the book so that I can get to the text instantly.

**User story:** As a user, I want access to additional information about the book so that I can read the book description and other meta data.

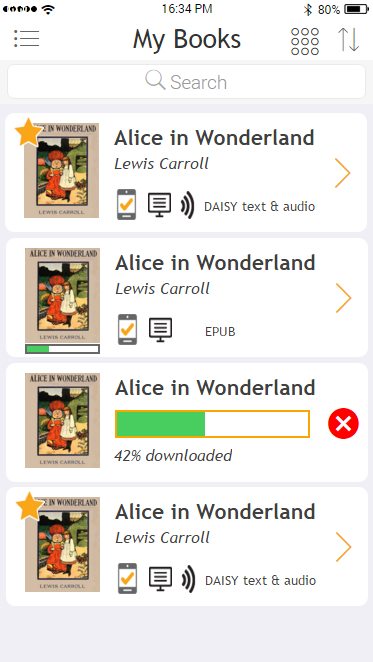
**User story:** As a user, I want to be able to search on title/author in My Books so that I easily can find the book I’m looking for.

**User story:** As a user, I want to be able to sort the books in latest, by title or author so that I can control the order of my books.

**User story:** As a user, I want to slide a book item to the left and get access to remove the book so that I can easily remove multiple books with a few gestures and taps.

This is the main screen which is shown after the splash screen and it contains the list of books that the user currently are reading.

*Definition of “currently are reading” = this is books that has been downloaded, or started to read in online mode. This means that online books that the user NOT has started to read, will not appear in this list.*



The toolbar contains this:

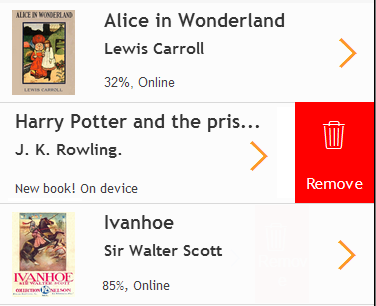
* Button to access main menu (see [Screen 10: Main navigation](#_Screen_10:_Main))
* Large title at the top “My Books”
* List/grid view button (see further down about grid layout)
* Sort button, which pops up a menu with options
  + Latest
    - This will sort the books in latest read order. Latest book at the top.  
      Note: Current downloads in progress should be visible at the top on “Latest”.
  + Title
    - This will sort the books by title, A-Z
  + Author
    - This will sort the books by author, A-Z

Next in the UI is a search bar, where the user can type in the search criteria. The search results needs to be presented in the same way as the book list.

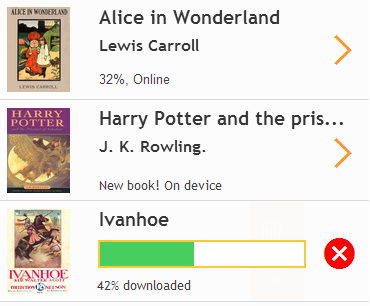
Last is a list of books and each book item needs to contain:

* Cover image
* Title
* Author
* Reading progress bar, under the cover image. Reading progress in the reader is a float value between 0 and 1 (0.22 = 22%).
* New status, of the book has no reading progress
* On device status (always visible in My Books, but for online search results it is only visible for content you already have in My Books)
* Text and/or audio images in book
* File format label

When sliding a book item to the left, you will get additional option to remove the book.



During download of a book, the book item needs to show a progress bar and an abort option similar to this:

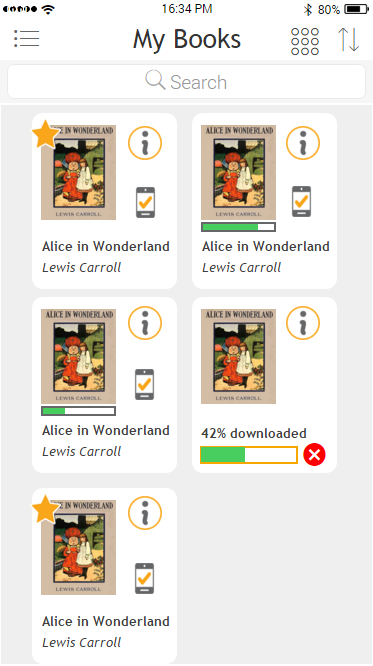


During a download, the book item must show download progress in form of a graphical progress bar and in text, e.g. “42% downloaded”. The user can abort the current download by pressing the abort button on the right hand side of the progress bar. EasyReader will ask the user: “Do you want to abort the download?”

If a user does a “long press” on a book item, we want the book actions to be shown. Please see section [Book actions](#_Book_actions) for more information.

### Show books in list and grid view

At the top of the “My Books” screen, there is a button that switches the list of books between list and grid view. The list view is explained above and this is how the grid view looks on a smaller device:



It basically contains the same information as the list view and share a lot of UI and behavior. However, the grid view items contains a bit less information simply due to the lack of space. A grid view item contains:

* Cover image
* “New” image (if book is new)
* Info button that connects to the book information screen
* “On device” image (if on device)
* Title label
* Author label
* Download progress with abort button is shown instead of the title and author labels

Please note that the number of grid view items in width (columns) are dependent on screen width. At minimum, we need to display two columns. Regardless of number of columns, the overall grid needs to be centered on screen (same amount of space to the left and to the right), however slightly smaller space between grid items similar to what the screen shot above shows.

When pressing on the info button, the book information screen needs to be displayed.

When pressing anywhere else on the grid item, the book will open in the reader. If the book is online (e.g. when the user are searching/browsing for online books), the press will open the book information screen. This is the exact same behavior as the list view of books.

Long press on grid view items should pop up the same list of options as in list view.

We can’t support slide-item-to-left to expose the “remove” option.

## Screen: Book information

**User story:** As a user, I want access to additional information about the book so that I can read the book description and other meta data.

**User story:** As a user, I want start a download (or play a book) from this screen so that I don’t need to do additional actions.

The book information screen contains the meta data information about the book and one or more actions the user can do.



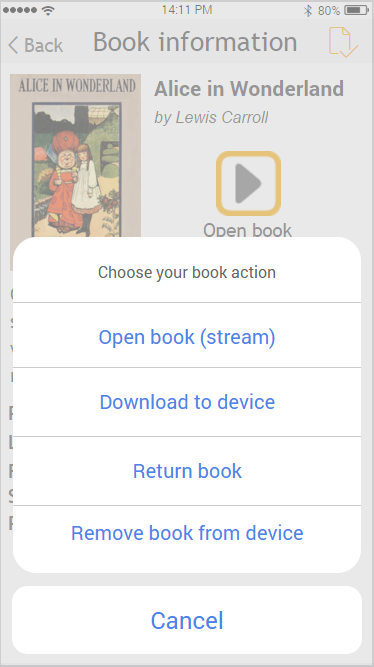
These are the controls in this screen:

* **Back** – this control takes you back to previous screen.
* **Screen title** – the top toolbar should contain a large “book information” text, so the user clearly knows where they are.
* **Book actions button** – pressing this one will give the user a full list of “book actions” (see below)
* **Cover image** – we need to display the book’s cover image here in a larger size than what’s shown in the book list.
* **Title & Author** – displays the title and author with the title larger and in bold, and the author in italic
* **Default action** – all books has a “default action” and we need to present this as a large icon/button and a short explanation text under it. The default action may e.g. be “Open book” or “Download”.
* **Description** – some books comes with a description text. As the text may be long, we need to fill the full width of the screen with the text. The meta data info will be pushed down when displaying long text.
* **Meta data** – each book comes with unknown number of meta data and these will be displayed in a long list.

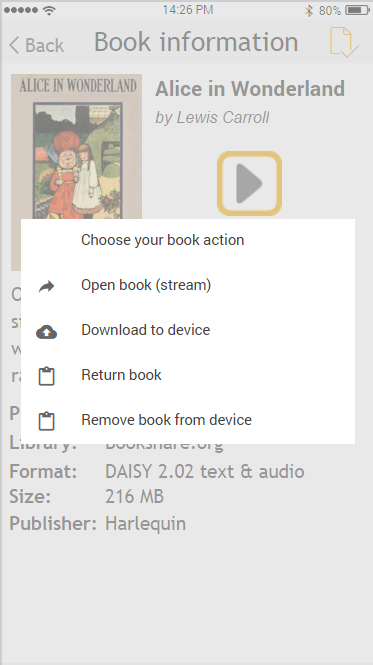
### Book actions

When pressing the “book actions” button at the top/right corner in the toolbar, a full list of actions is displayed. The default book action is always at the top and then the other actions follows. The list of actions will be different if the book is online or downloaded and from which library it comes from.

For iOS it will look like this:



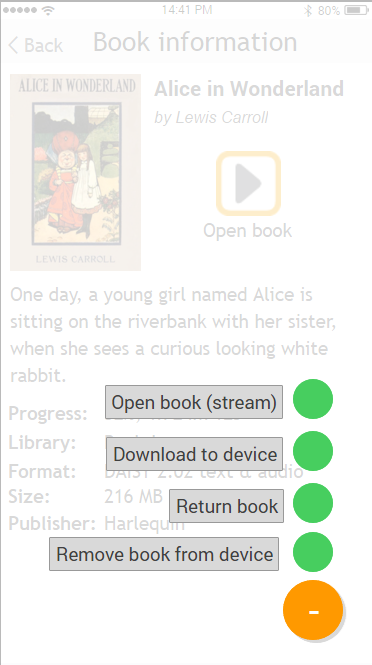
For Android, we are considering two options. The first one is more or less similar to the iOS solution, but in the center of the screen:



Or on Android, we can use the floating action buttons with the floating book action button at the bottom of the screen:



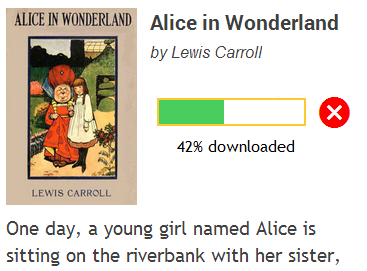
When pressing that button, more floating buttons will appear in an animated way and it also fades out the background:



The prototype screen shot above is a bit ugly ☺ but you can see different samples here:

<https://www.google.se/search?q=android+floating+action+button&espv=2&biw=1378&bih=764&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjY3ayfrZ3MAhXlB5oKHbz2BEkQ_AUIBygB&gws_rd=cr&ei=WssYV5vfIeKP6ASkuKSwBA#imgrc=_>

If a book currently is being downloaded, the area for the “default action” should instead show the current download progress with an abort button.



When pressing the “abort” button, EasyReader need to ask the user if abort is going to be made or not.

After a completed download, it will shorty say “Download compete” with 100% filled progress bar (for 0.5 to 1 seconds?). Then replace with the “open book” book action.

## Screen: Reader

**User story:** As a user, I want the book to open from last known position so that continue to read from where I last read.

**User story:** As a user who sometimes get books from different libraries, I want open and read all my books irrespective of file type so that I can just read without worrying about where the book come from or who made it accessible.

**User story:** As a student reading a book, I want to make notes in the book so that I can use my notes as part of my study technique.

**User story:** As a book reader, I want to easily skip forward and back through the text so that I can locate the parts I’m most interested in and skip the boring parts.

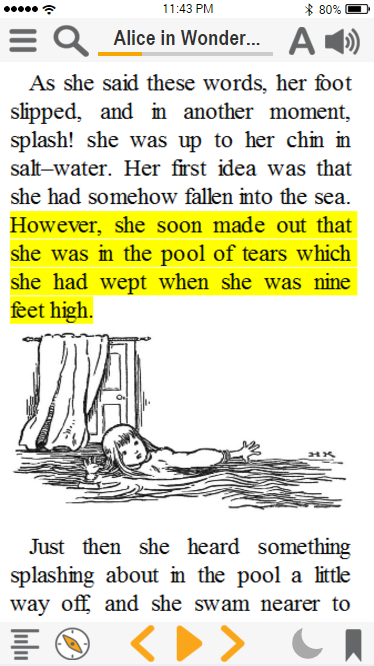
**User story:** As a partial sighted user, I want enlarge the text and make sure the text wraps so that I can read the words on screen without scrolling/panning sideways.

**~~User story:~~** ~~As a user, I want to hide the top and bottom toolbars so that I can maximize the reading area.~~

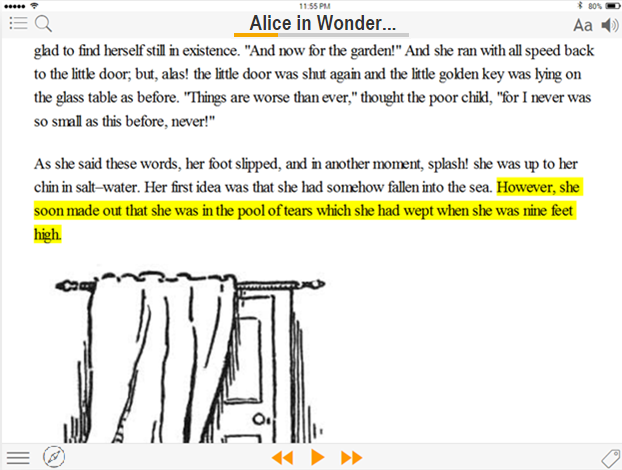
**User story:** As a user, I want single tap the text for navigation so that I don’t need to use multiple other navigation methods to get to that point.

**User story:** As a user, I want double tap the text to start and stop playback so that I don’t need to locate the play/stop button to do the action.

On a small screen (phone), the reader looks like this:



On a larger screen (table), the reader look like this in landscape mode:



The top toolbar contains this:

* **Menu button**Opens the main slide menu
* **Search button**Opens [a search screen](#_Search_in_book) where you can type in the search criteria and the results are presented in a list.
* **Title label with current book position**Displays the book title and a progress bar with the current reading position.   
  If the user taps the label, the user will get more information about the current position and ability to navigate by percent/time (see further down for more information).
* **Text settings button**Pops up a window that displays a text settings [Reader text settings](#_Reader_text_settings)
* **Audio settings button**Pops up a window that display a settings [Reader audio settings](#_Reader_audio_settings)

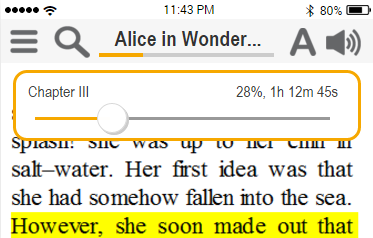
The middle of the screen is where the content is displayed. Few behaviors:

* **Single tap** = navigate if the tap is in text
* **Long tap** = select current word and pop up the toolbar
* **One finger swipe up/down** = scrolls the text (default behavior for web view)
* **Two finger pinch** = change font size larger/smaller
* **Two finger double tap** = start/stop playback
* **Two finger swipe left/right** = does the prev/next buttons
* **Two finger swipe up/down** = up for increasing the audio speed, down for decreasing audio speed

The bottom of the screen contains this:

* **Book navigation**This goes to the screen [Screen: Book navigation](#_Screen:_Book_navigation)
* **Navigation mode**This screen goes to the screen [Screen: Navigation modes](#_Book_navigation_modes)
* **Navigate previous**Navigates to the previous, defined by the current navigation mode
* **Play/Stop**This button starts or stops the playback of the book.
* **Navigate next**Navigates to the next, defined by the current navigation mode
* **Sleep timer**  
  This button opens [the sleep timer options](#_Sleep_timer).
* **Bookmark**This button opens the create/edit notes: [Create note](#_Screen:_Create_note)  
  *To be discussed: how do we visually show a bookmark on screen? Similar to what we do in ER 7 for Windows?*

The user can single tap the label at the top of the screen and a small popup screen with a slider and some other information will be shown.



This popup contains:

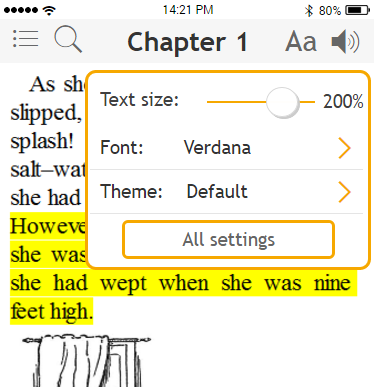
* **Label (left)** – contains the current heading name
* **Label (right)** – contains the current position in percent. If the book has audio, current time in the book will be presented.
* **Slider** – this is for navigating in the book by percent.
  + When sliding the button, we need to update the percent (and time) information in the reader. Note: no active navigation on the fly.
  + When releasing the button, we will navigate to the new percent value and the reader will navigate to the new position. Keep this window still open until the user taps outside it.

This popup can be closed by tapping outside the window, but it will also automatically close after 4 seconds. If VoiceOver is active and in focus in the popup, the popup shouldn’t automatically close that quick (perhaps max 30s).

### Text settings

The user can adjust text settings on the fly and it will be applied directly into the content area of the reader.

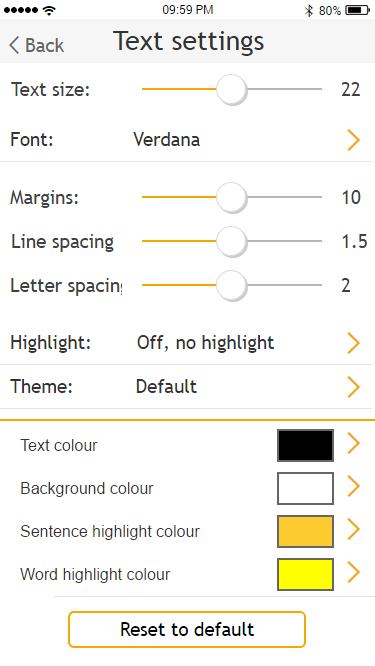
**Note**: if the user have a small device (phone), we need to display a pop up window with a sub set of settings. These are the settings that most likely are adjusted most. To get access to all text settings, press the “all settings” button.  
If the user have a larger display (tablet), it goes directly to show all text settings. On smaller screens, this is a separate screen, but on larger screens all settings will be displayed in a pop up window. See further below.



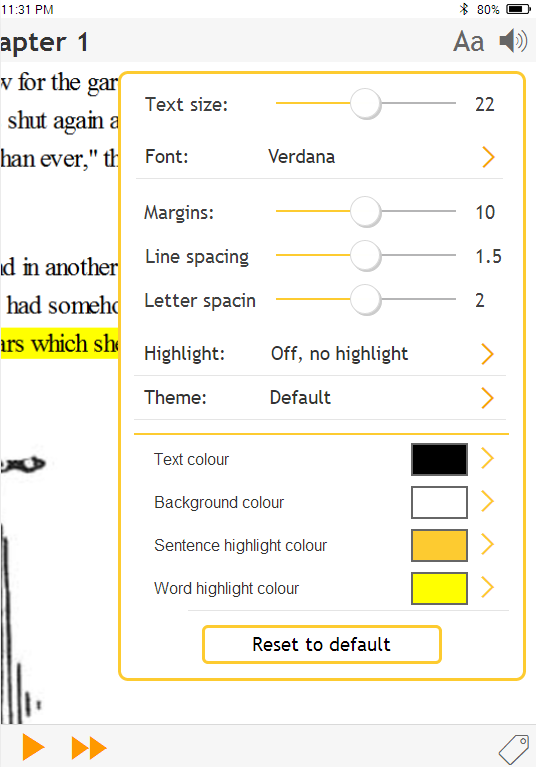
These are the controls:

* **Text size slider**Increases or decreases the font size of the text.
* **Font drop down**The user can choose to apply a font in the list, or select the first item “Book’s font” to NOT apply a font setting to the content.
* **Theme dropdown**The user can choose from a list of color themes in a similar way as EasyReader for Windows. Options are:
  + Default
  + Black
  + Dark Blue
  + Cream
  + Light Blue
* **All settings button**Pressing this button displays a screen with additional text settings. See below

On smaller screens (phone):



On larger screens (tablet):



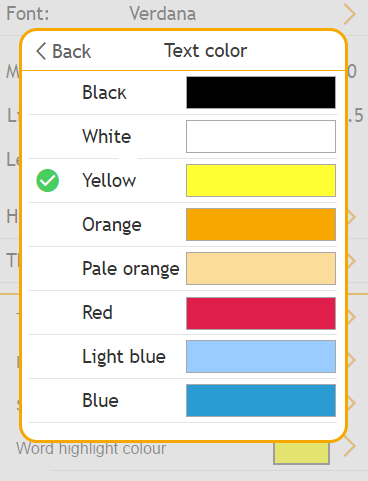
These are the controls and settings:

* **Text size**On EasyReader for Windows, this setting actually is a zoom feature in the web browser control, as we want all images, tables and other things to increase or decrease in size naturally.  
  Research on Android/iOS: can we use the web browser’s zoom feature for this?  
  To be decided: Values of the slider and what to write out as text on the label.
* **Font**The user can either use the default font on the system or from a list of dyslexia friendly fonts that EasyReader has embedded
  + Arial
  + Calibri
  + Century Gothic
  + Comic Sans
  + Open Sans
  + OpenDyslexic
  + Quicksand
  + Tiresias
  + Trebuchet
  + Verdana
* **Margins**The user can apply a margin to the document and available settings are:
  + 0% (margin = off)
  + 3%
  + 6%
  + 10%
  + 15%
  + 20%
* **Line spacing**The user can adjust the line spacing in the document and available settings are:
  + None
  + 125%
  + 150%
  + 200%
  + 250%
  + 300%
* **Letter spacing**The user can adjust the letter spacing in the document and available settings are:
  + 0pt (off)
  + 1pt
  + 2pt
  + 4pt
* **Highlight setting**With this setting, the user can control to show/hide the highlighter etc and available settings are:
  + Off (no highlight)
  + Word
  + Sentence
  + Sentence and Word
* **Theme setting**EasyReader comes with a number of predefined color sets and they are listed in the table in section [Colors and themes](#_Colors_and_themes). When the user choses another color (e.g. highlight color), it will save it into the chosen theme. If the user want to go back to original theme colors, the “reset to default” button needs to be pressed.

NOTE: RGB values are defined in the section [Colors and themes](#_Reader_audio_settings).

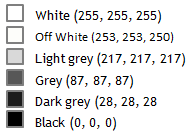
* **Text color**There are four different text color settings and all of them opens a color picker control.
  + **Text colour** – sets the document text colour.
  + **Background colour** – sets the document background colour.
  + **Sentence highlight colour** – sets the current sentence highlight background colour
  + **Word highlight colour** – sets the current word highlight background colour
* **Reset to default**This is a button that resets all settings in “text settings” to its default values. A confirmation dialog must be shown first asking the user to proceed to not.

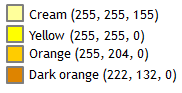
The colors available in the picker view window are defined in the section [Colors and themes](#_Reader_audio_settings). The color picker may look similar to this:

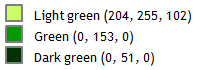


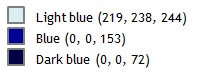
### Colors and themes

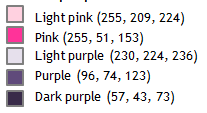
The colors available in the color picker view are these:

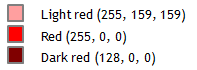


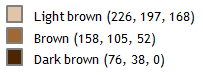








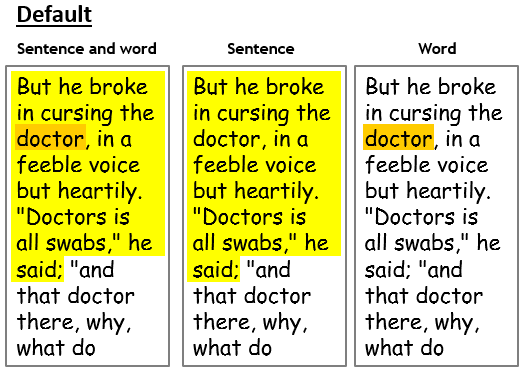


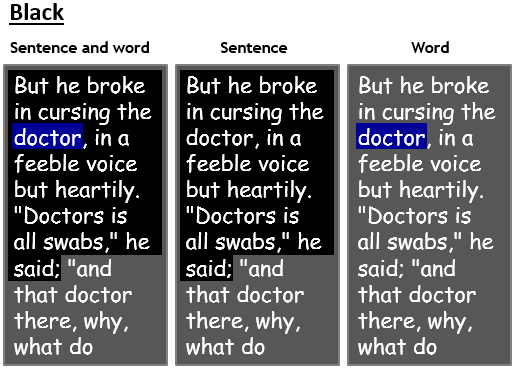


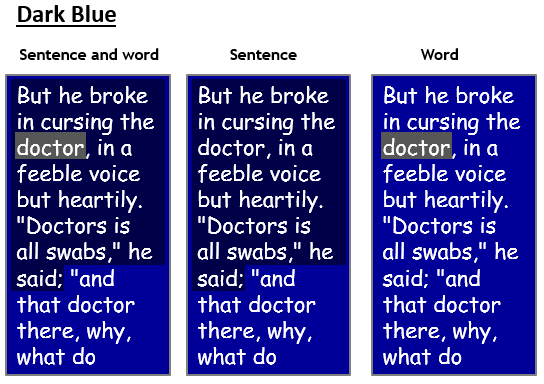
The themes are its colors are defined here:

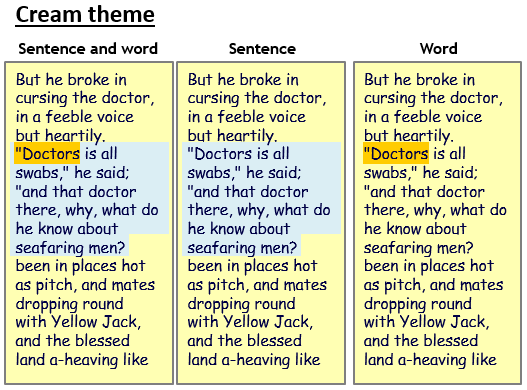
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Background** | **Text** | **Sentence highlight** | **Word highlight** |
| Default | White (255,255,255) | Black (0,0,0) | Yellow (255,255,0) | Orange (255,204,0) |
| Black | Grey (87,87,87) | White (255,255,255) | Black (0,0,0) | Blue (0,0,153) |
| Dark blue | Blue (0,0,153) | White (255,255,255) | Dark Blue (0,0,72) | Grey (87,87,87) |
| Cream | Cream (255,255,155) | Dark Blue (0,0,72) | Light Blue (219,238,244) | Orange (255,204,0) |
| Light blue | Light Blue (219,238,244) | Dark Blue (0,0,72) | Cream (255,255,155) | Orange (255,204,0) |

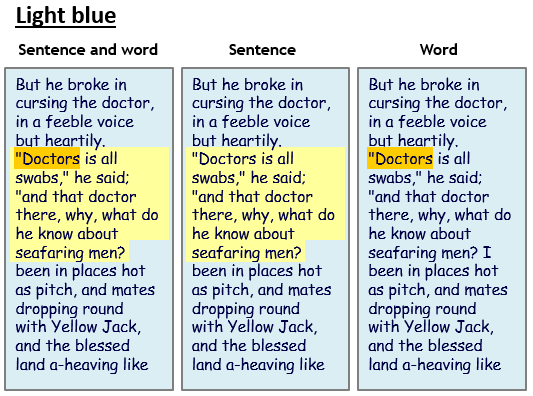
Below is some sample screen shots how the themes may look like in web view control in the reader.







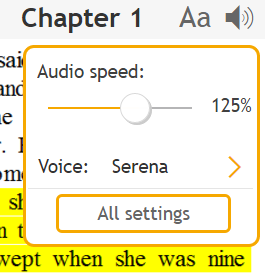




### Audio settings

The user can adjust audio settings by pressing the audio settings button.

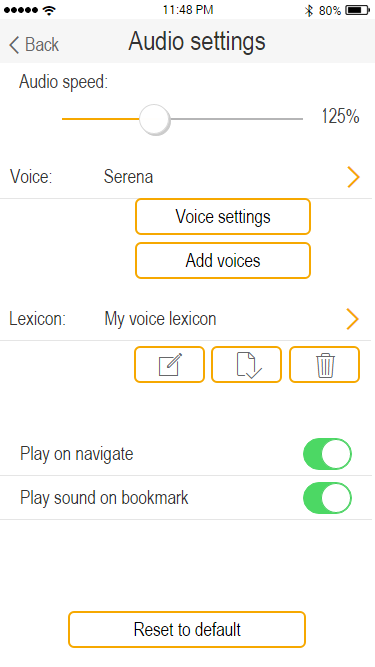
**Note**: if the user have a small device (phone), we need to display a pop up window with a sub set of the audio settings. These are the settings that most likely are adjusted most. To get access to all audio settings, press the “all settings” button.  
If the user have a larger display (tablet), it goes directly to show all audio settings. On smaller screens, this is a separate screen, but on larger screens all settings will be displayed in a pop up window. See further below.



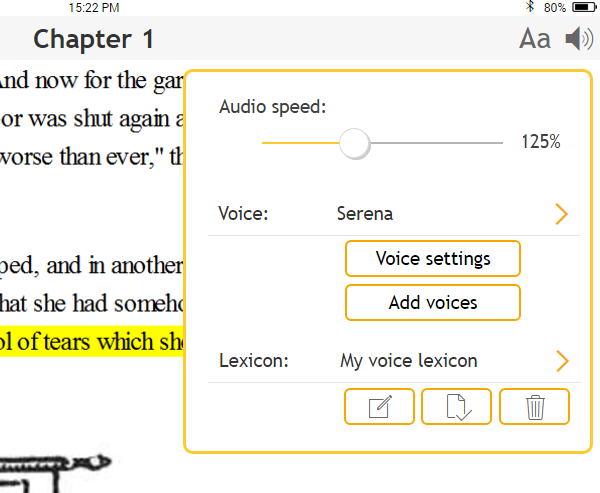
These are the controls:

* **Audio speed** – controls the setting for speed on both TTS voice and MP3 audio
* **TTS Voice** – a dropdown control with the available voices. The last item is “Add more voices”, which goes to the screen for In-App purchases of voices: [In-App purchases of TTS voices](#_In-App_purchases_of)
* **All settings button** – pressing this opens the screen for all audio settings (see below).

On smaller screens (phone):



On larger screens (tablet):

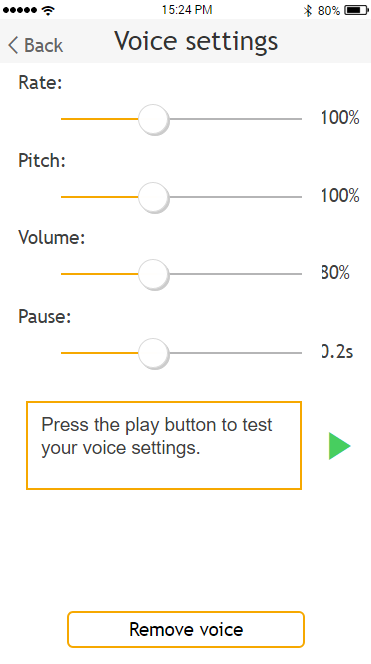


These are the controls and settings:

* **Audio speed** – controls the setting for speed on both TTS voice and MP3 audio.  
  For MP3 audio, the label need to write out the speed in %, e.g. “150%”  
  For TTS audio, the label need to write out the speed in words per minute (wpm), e.g. “180 wpm”
* **Voices list** – a dropdown control of all available voices. If no voices are available on the system, this dropdown control needs to be disabled. In addition, the “Voice settings” button need to be disabled.
* **Voice settings button** – this button opens the [screen for TTS settings](#_Voice_settings)
* **Add voices button**  - this button goes to the screen for In-App purchases of voices in the [screen Add Voices](#_Add_voices).
* **Voice lexicon** controls (disable all these controls, if no TTS voices are available on the device):
  + **Dropdown list –** a list of all lexicons
  + **New button –** creates a new lexicon and the user need to type in a new name first and then it opens the screen [Screen: Voice lexicon](#_Screen:_Voice_lexicon)
  + **Edit button** **–** opens the screen [Screen: Voice lexicon](#_Screen:_Voice_lexicon)
  + **Remove button –** pops up a question “Do you want to remove the selected lexicon?” and will remove the lexicon if the uses chooses “Yes”.
* **Play on navigate** – if this is ON, the reader will automatically start play back when:
  + Open a book
  + Navigate by heading
  + Navigate by page
  + Navigate by bookmark
  + Navigate next/prev
  + Navigate on search word
* **Play sound on bookmark** – if this is ON, the reader will play a short audio clip when reaching to a book position where a bookmark exist. Please note that this short audio clip can be played at the same time as the book’s audio.
* **Reset to default**This is a button that resets all settings in “audio settings” to its default values. A confirmation dialog must be shown first asking the user to proceed to not. Default values:
  + Audio speed to 100%
  + Voice name to <TO\_BE\_DEFINED> What is our default voice on app startup?
  + Lexicon, select the first user created lexicon

### Voice settings

This screen is for adjusting the TTS settings on the Nuance voices. The settings are applied across all Nuance voices (not saved per voice).

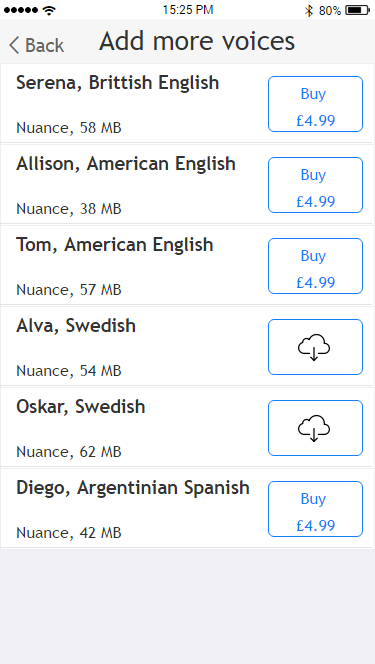


This screen contains:

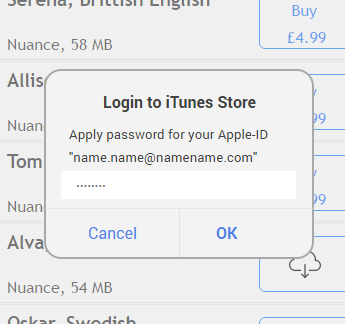
* **Label “Rate”, slider and value label** – ranging from 50 to 400 with default value of 100. The value label must express the value in percent, e.g. “150%”
* **Label “Pitch”, slider and value label** – ranging from 50 to 200 with default value of 100. The value label must express the value in percent, e.g. “150%”
* **Label “Volume”, slider and value label** – ranging from 0 to 100 with default value of 80. The value label must express the value in percent, e.g. “80%”
* **Label “Pause”, slider and value label** – ranging from 0 to 9 with default value of 1. Each unit/step has the value of 200ms of silence, so if the value is 2, then the value label must say “0.4s”.
* **Text edit area and play button** – the edit area must contain a default text which is spoken out when pressing the Play button. The edit area is editable, but there is no need to save the text in app storage. There is no need to switch icon from “play” to “stop” and vice versa.
* **Remove voice button** – pressing this button will first ask the user “Do you want to remove the voice <ADD\_VOICE\_NAME\_HERE> from this device?”. On confirmation, the app will remove the voice from the device and return to the Audio Settings screen.

### Add voices and In-App purchase

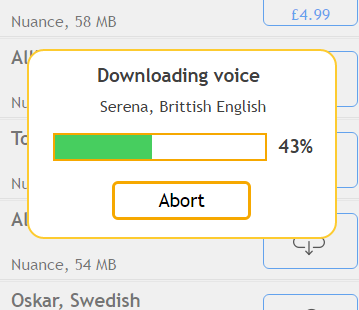
To be added…



Dasd



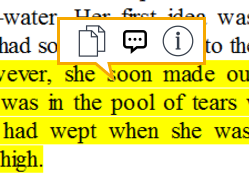
Dasdas



dasd

### Reader popup toolbar

When selecting a word in the content area, the app selects the text under the finger and this popup window is displayed. Please note that the user can at this point expand the text selection freely.



This toolbar area contains this:

* **Copy text button**This button will copy the selected text to the device’s clipboard. The text can then be imported in “My Text”, or used in a different app on the device.
* **TTS lexicon button**This opens the screen for [Screen: Voice lexicon](#_Screen:_Voice_lexicon)
* **Word lookup button**This opens the screen for [Screen: Word Lookup](#_Screen:_Word_Lookup)

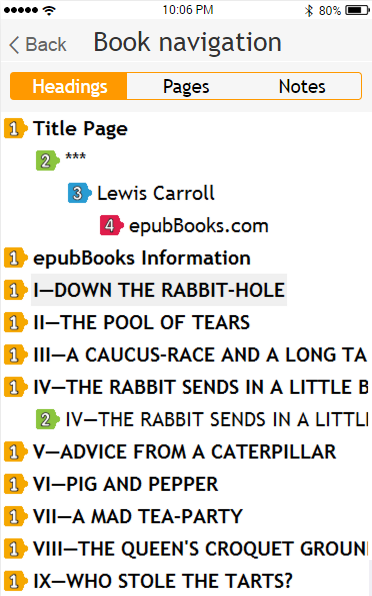
### Book navigation

**User story:** As a user, I want have easy access to book navigation so that I don’t need to do many taps on screen.

**User story:** As a user, I want to navigate by heading/page/notes so that I quickly can navigate in the book.

**User story:** As a user, I want to be able to type in a page number to search/navigate to so that I don’t need to scroll a very long list of page numbers.

This screen contains the main navigation methods the user can do within a book and the screen is split into three tabs; “Headings”, “Pages” and “Notes”.

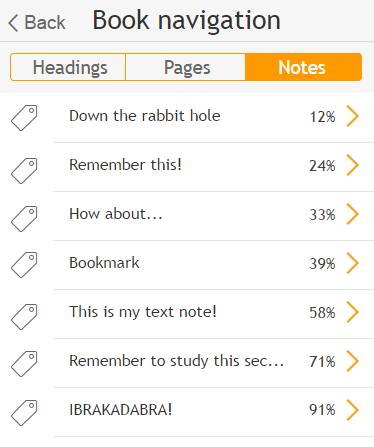


The first tab “Headings” contains all navigable headings in the book. Each heading should contain an icon representing the level and level 2 to 6 should be indented. When tapping a heading, this screen closes and navigates to the selected heading.

The second tab “Pages” contains all navigable pages in the book (if such exists).



This tab has a search edit area so that the user can type in a page number to search/navigate to. It also contains information about the current page number based on the current position in the book. Last, it has a list of all pages in the book, where the current page number is preselected and visible. When tapping a page number, it will close this screen and navigate to that page in the book.



This tab “Notes” contains all bookmark and notes the user has done in the book and all of them are listed in reading order. When the user selects an item in the list (tap), this screen will close and the reader navigates to the position of the bookmark/note.

However, if tapping at the end (at the orange > icon), the screen for editing the notes will open: [Create note](#_Create_note). Pressing back from that screen goes back to this list of bookmarks and notes.

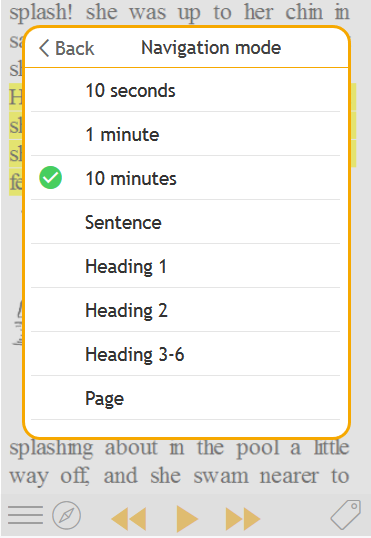
The user can delete a bookmark by swiping the item to the left and choose delete. Very same as for deleting books and unsubscribe to newspapers.

When entering this book navigation screen, it need to show the last used tab. The reason is to simplify usability when e.g. navigating notes by notes and this would save an extra tap on the screen. When loading a book, it needs to be reset and set to default “headings” tab.

### Book navigation modes

**User story:** As a user, I want adjust the amount of navigation for prev/next so that I can be in better control of how to navigate around the book.

The navigation modes controls how much the prev and next navigation buttons navigate in the book and when pressing that icon/button in the reader view, this screen is shown:



The settings are:

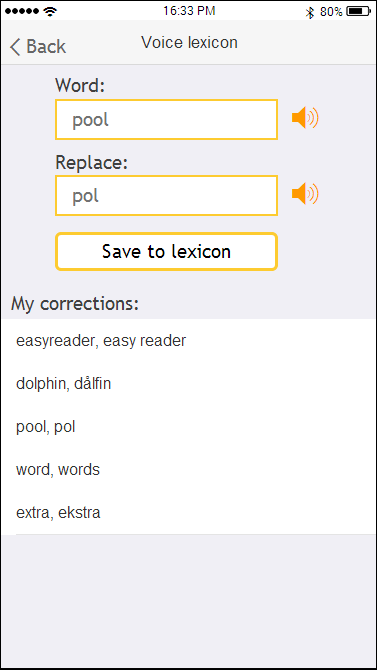
* **10 seconds** – navigates 10 seconds in a book with audio
* **1 minute** – navigates 1 minute in a book with audio
* **10 minutes** – navigates 10 minutes in a book with audio
* **Word** – navigates word by word in a book with text
* **Sentence** – navigates sentence by sentence in a book with text
* **Page** – navigates by pages in a book with pages
* **Heading 1** – navigates by heading level 1 in a book with headings
* **Heading 2** – navigates by heading level 2 in a book with headings
* **Heading 3 - 6** – navigates by heading level 3 to 6 in a book with headings
* **Bookmark and notes** – navigates by bookmark or notes if such exists

If the currently loaded book type doesn’t support any of these above, items should be greyed out and disabled.

When the book loads, the reader will check the validity of this user settings against the loaded book. If it can’t match, this setting will default back to “sentence” for a book with text and to “1 minute” for book with audio.

### Voice lexicon

To be added…

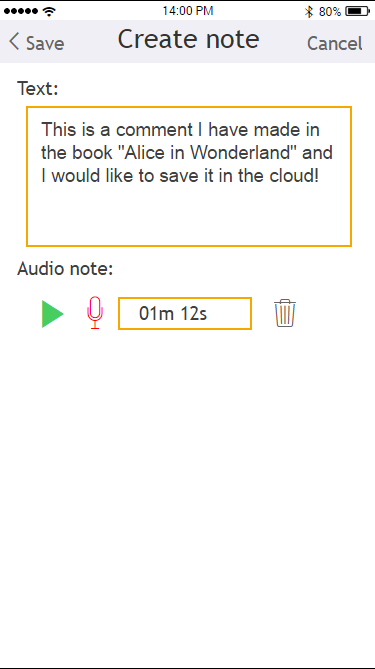


To be added…

### Create note

**User story:** As a student reading a book, I want to make notes in the book so that I can use my notes as part of my study technique.

When adding a note (or editing an existing note), this screen will appear:



This screen contains:

* Save button
* Title with “Create note” or “Edit note”
* Cancel button
* Text label + edit area (max 200 characters?)
* Audio note label
* Play button
* Record button (max 60 seconds?)
* Time label (for recording + indicated audio note length)
* Delete button (only visible if audio note exist)

All bookmarks are saved on the device storage and each platform needs to write its own handling for this. You need to retrieve from the reader module the current ID of the current reading position and use that for the identifier for the bookmark. You can also retrieve the text from the ID, to automatically populate the text edit area with some default text.

If the reader’s current ID already have a bookmark on the device storage, this would become editing of an existing bookmark. The already saved text and/or audio notes needs to be populated into this screen.

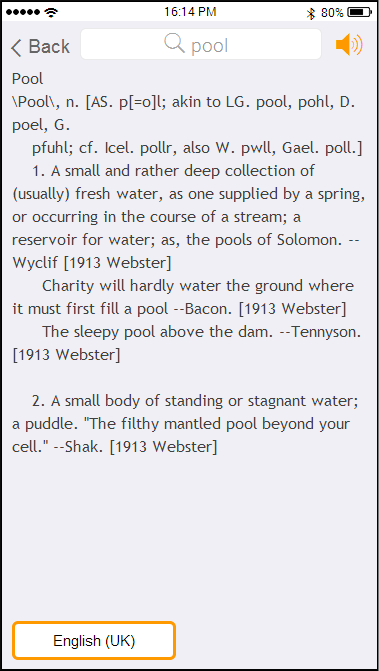
**To be defined**: we need to somehow indicate the existence of a bookmark in the reader view. We could e.g. use the bookmark ID and the JavaScript to show a graphical “thing” near the ID element by using relative positioning of the graphics. Or put it on the far left side of the screen (on the same row as the ID element).

**Later in the project**: these stored bookmarks needs to be synchronized with the Dolphin data storage web service.

### Word Lookup

**User story:** As a reader who encounters an unfamiliar word, I want look up the definition of the word so that I can understand what the word means.

To be added…

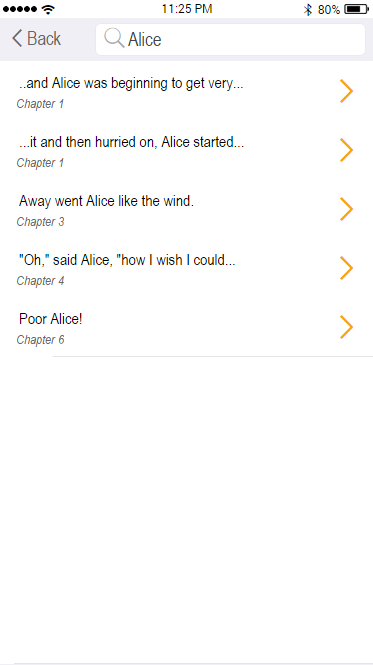


To be added…

### Search in book

**User story:** As a user, I want quick and easy access to the search feature in the reader, so that I easily can find and navigate to the word I’m looking for.

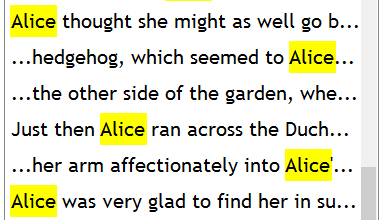
When the user taps the “search” button in the reader, this screen is shown and focus is automatically set into the search edit area.



This screen contains:

* Back – for navigating back to the reader
* Search edit area – the user types in the search word here
* Search result list – the list is populated with the search result list from the whole book and each item contains:
  + A piece of text containing the search word highlighted. If the text is too large to fit into the available space, the text needs to be truncated so that the search word is visible and highlighted.
  + Current heading name in italic (could also be slightly more grey in color, if the search text is black)
  + Icon > for indicating an action

NOTE: The prototype screen shot doesn’t show the highlight of the search word, but we want a similar look as in EasyReader for Windows:



When the user taps on a search item, this screen closes and the reader will navigate to the search word.

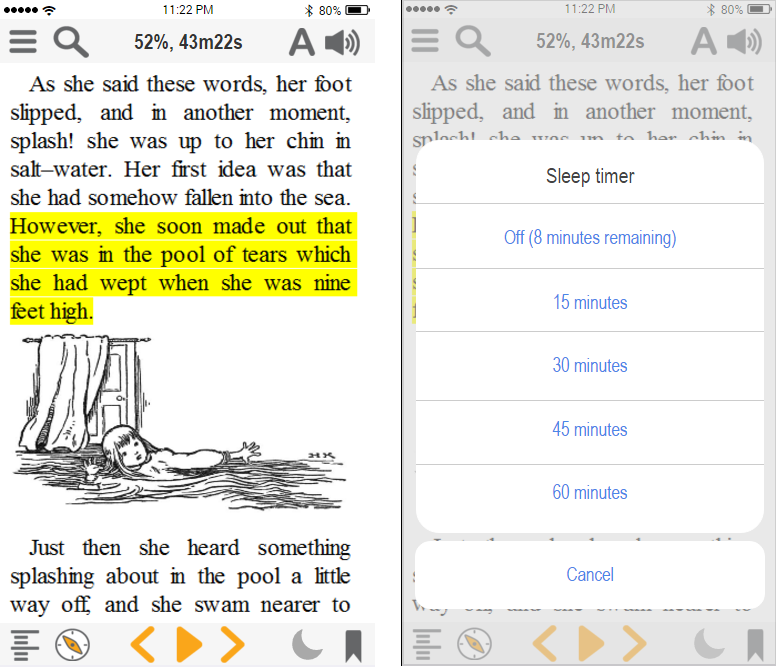
When returning back to this screen, it would be nice if the earlier made search word and search results already are populated on screen. Please remember to clear this search data when loading a new book.

On iPad (tablet): You can use the new popup window for this search screen.

### Sleep timer

**User story:** As a user, I want to use a “sleep timer” with automatic stop, so that the playback doesn’t go to the end of the book when I fall asleep in the bed.

A sleep timer is quite common for audio-enabled book reading apps and we need to have one as well. The reader have a button next to the bookmark button and it show the sleep timer options similar to below.



By choosing one of the values (15, 30, 45 or 60 minutes), it will trigger a timer in the reader that will do an automatic stop of ongoing playback at the end of this timer. If the reader already is in stop mode, the timer simply quits with no action. If the reader is in play mode, the playback stops. Please note that the user can do play, stop, navigate, settings changes and so on during an active sleep timer. The sleep timer also need to work when the app is in the background or on the locked screen.

If the sleep timer is active, the OFF menu item needs to display the remaining time. Please note that this doesn’t need to be updated “live” as long as the menu is open. Just grab the remaining time when showing this menu and display it.

The sleep timer needs to be turned off in these situations:

* Pressing OFF in this menu
* Closing the book
* Closing the app

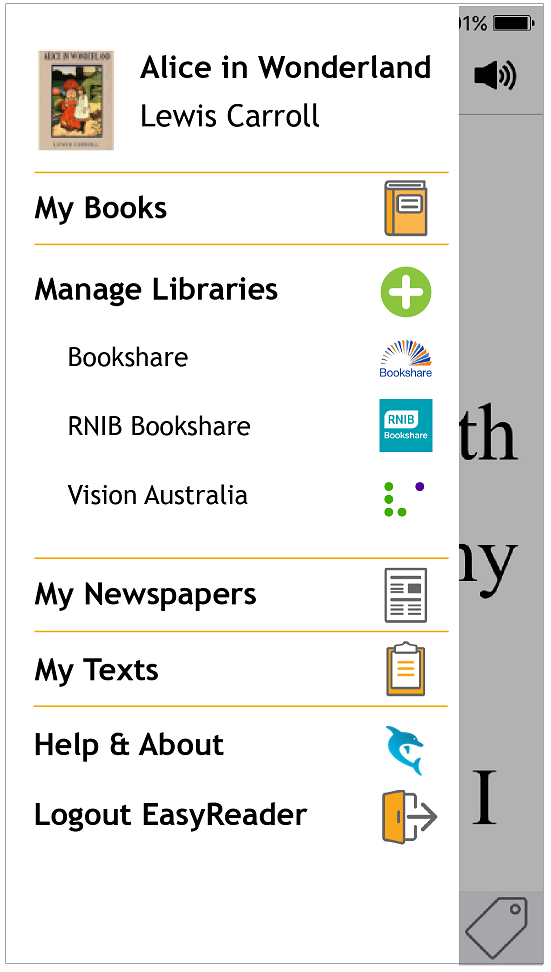
## Screen: Main navigation

**User story:** As a user, I want to access to my libraries within a few taps, so that I quickly can search and browse for new books.

**User story:** As a user, I want quickly be able to access my newspaper subscription so that I can read the daily news.

This is a slide menu that slides in from the left side of the screen and do not cover all the screen. This is the main navigation panel to access different libraries, newspaper subscriptions, settings, etc.

To close this menu, you either choose one of the menu items, or slide the menu back to the left side of the screen.



Items in the slide menu:

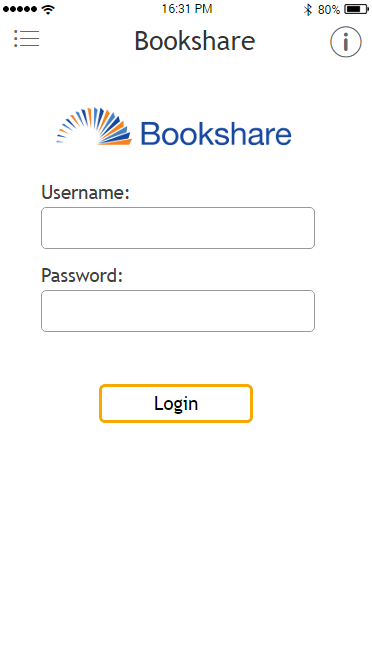
* Latest read – a quick link to the latest read book, or the book which currently is loaded in the reader
* My Books – this accesses the screen [Screen 2: My Books](#_Screen_2:_My)
* Manage libraries – this goes to the screen [Screen 14: Manage libraries](#_Screen_14:_Manage)
* Book libraries – this is a dynamic list of preferred book libraries, that the user sets up in [Screen 14: Manage libraries](#_Screen_14:_Manage). When pressing on a book library item, it goes to the screen [Screen 12: Browse categories](#_Screen_12:_Browse) or [Screen 13: Search results](#_Screen_13:_Search), dependent on the type of library. See bookshelf module documentation for more information.
* My Newspapers – this accesses the user’s subscribed newspapers in screen [Screen 16: My Newspapers](#_Screen_16:_My)
* My Text – this is the screen [Screen 17: My Texts](#_Screen_17:_My) where the user can access saved texts in the cloud or to import/paste new text.
* Help & About – goes to the screen [Screen 19: About](#_Screen_19:_About)
* Logout EasyReader – logs out the user from EasyReader and goes to [the Getting Started screen](#_Screen:_Getting_started).

Please note that the book libraries are dynamic menus (can be zero to many items) based on user settings. All other items can be hard-coded into the menu.

Please also note the design of the side menu with the spacing between items, bold and no-bold text, orange lines and icons on all items to the right side of the menu.

## Screen: Login

The login screen is displayed when accessing libraries which requires user authentication. Please note that this screen will be skipped if the login credentials are saved in memory or settings.



Screen items:

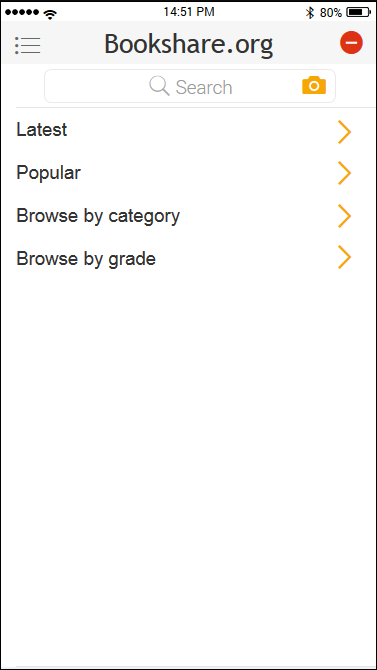
* Main slide menu at the top/left corner
* Label centered at the top with the library name
* Info button that goes to the [library information](#_Screen:_Library_information) screen.
* Library logo (with library name as accessibility text for VoiceOver/TalkBack)
* Username label and edit area
* Password label and edit area (which prints out \* for each character)
* Login button

On failed login, we need to alert the user with “Username or password is incorrect. Please try again.”

On successful login, the credentials are automatically saved for future automatic logins and then next screen is displayed. Dependent which library it is, it goes to [Screen: Browse categories](#_Screen:_Browse_categories) or directly to [Screen: Search results](#_Screen:_Search_results).

### Screen: Browse categories

To be added…

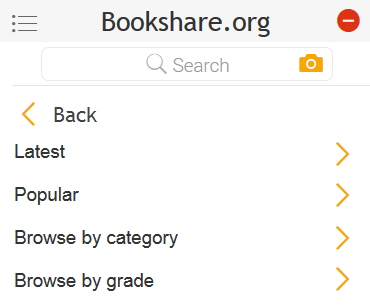


Controls in this screen:

* **Main menu** – access to the main slide menu [Screen: Main navigation](#_Screen_10:_Main)
* **Library title** – the name of the library
* **Log out button** – when pressing this one, the screen for login will be shown.  
  Please note that this button is only visible for libraries with login requirement.
* **Search edit area** – the user can type in a search request and the resulting books will be presented in [Screen: Search results](#_Screen_13:_Search)  
  The camera icon/button is right now optional for bar code scanner search
* **List of categories** – these are the library categories sent from the online servers via the bookshelf module.

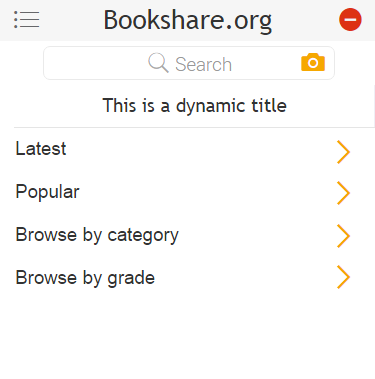
Please note that the search feature is not available for all libraries, so this one needs to be hidden those libraries. This is information available from the bookshelf module.

When using multiple layers of categories, we need to make sure the user can navigate back to previous categories. We will use the same method as we do in EasyReader v7 and add a “Back” menu item if the UI is presenting category level 2 or deeper. Similar to this:



For each layer of categories, the apps need to check if search is available or not and hide/show the search feature.

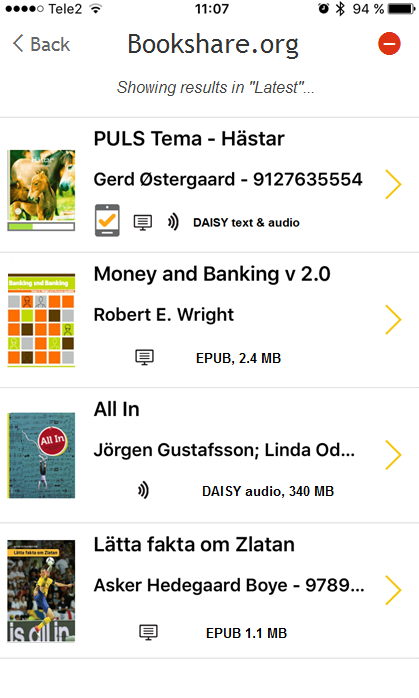
Some online libraries also includes a title on some category menus and in that case, we will add a label area below the search box (if it is present) and above the list of categories.



This currently may appear on DZDN and Passend Lezen.

### Screen: Search results

To be added…



The toolbar area contains this:

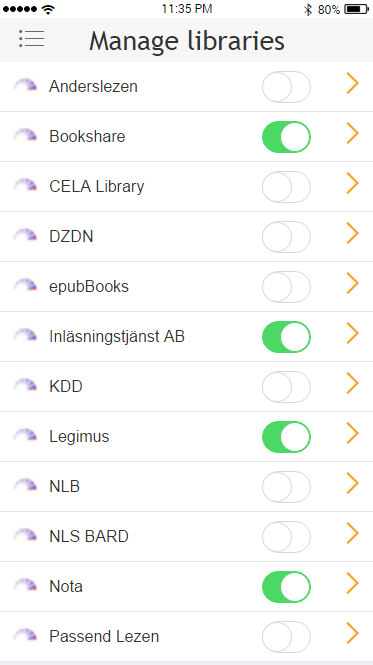
* Back button
* Title label, which contains the library name
* Logout button (if pressing it, we should ask the user “Do you want to log out from <LIBRARY\_NAME>?”
* Label stating “Showing results from <CATEGORY\_NAME>” (if applicable)
* If library supports it, this screen may also show an search edit area located just above the book list (same as for My Books)
* List of book results

The book results will contain a list of books very similar to what’s in [My Books](#_Screen:_My_Books). Please note that online search results need to indicate clearly if the book already has been downloaded or not. An already downloaded book has new or reading progress and an “on device” image. If the book is online, it will not have “new” image, not a reading progress and not a “on device” image. The only difference is that we add the size of the book to the end of the file format label.

## Screen: Manage libraries

**User story:** As a user, I want select my preferred libraries so that I don’t need to navigate a long list every time.

This screen contains all available libraries, easy way to turn a library on or off and the user can also tap an item to get more information about the library. The order of libraries is in alphabetic order.



Each list item contains:

* Icon image of the library. This small icon is available through the bookshelf native module interface.
* Library name
* On/Off switch, which controls if the library is available in the side menu or not.
* Icon indicating there is more information about the libray, see next section: [Screen: Library information](#_Screen:_Library_information)

For screen readers on the apps, it is important that it announces the library name, how to switch on/off and that more library information is available when tapping the item.

### Screen: Library information

**User story:** As a user, I want read about the library so that I know what type of content the library have before accessing it in the app.

This screen have basic information about the selected library.

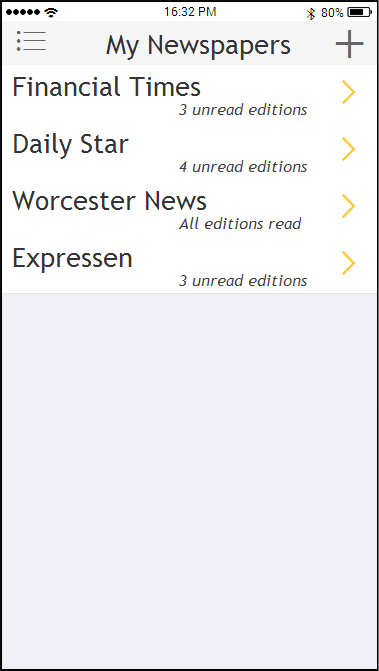
This screen contains:

* Library name in the toolbar area
* Library logo
* Library information (plain text, not HTML)
* Link to the library web site

## Screen: My Newspapers

**User story:** As a user, I want quickly be able to access my newspaper subscription so that I can read the daily news.

By selecting “My Newspapers” in the main navigation menu, a screen opens which handles the user’s newspaper subscriptions. It is important the behavior is as similar as possible to what’s available in EasyReader v7 for Windows.



The top toolbar contains:

* Navigation button – opens the main navigation menu [Screen: Main navigation](#_Screen:_Main_navigation)
* Title label – just a large label with the text “My Newspapers”
* “Plus” button – pressing this button will open up a new screen (see below) for adding more newspapers to the subscribed list.

The list contains all subscribed newspapers and when tapping on a newspaper item, the next screen is opened (see below)

The user can unsubscribe to a newspaper by sliding the item to the left and choose “Unsubscribe”.

### Screen: Newspaper editions

This screen contains all the latest editions for the selected newspaper in “My newspaper”.



The top toolbar contains:

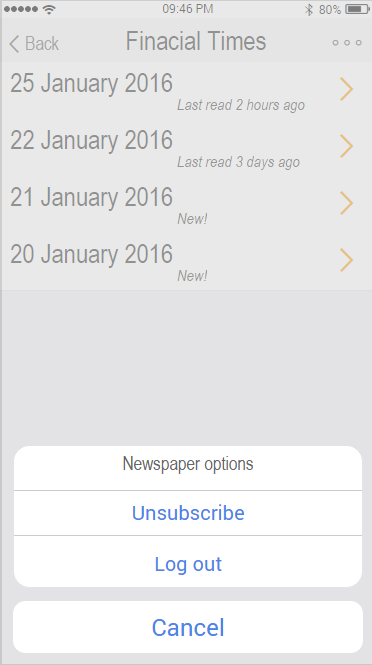
* Back – pressing this goes back the main menu for newspapers (see above)
* Title label – the large label need to contain the name of the selected newspaper
* “More…” button – a button for extra actions (unsubscribe and log out)

When tapping on a newspaper edition, it will be opened in the reader: [Screen: Reader](#_Screen:_Reader)

Note: when returning back from the reader, it will return back to “My Newspapers” top menu (not the editions). Similar is for “My Books” and “My Texts”.

The user can delete a read newspaper edition by sliding the item to the left and choose “Remove”.

When tapping the “more…” button at the top right corner, this screen is shown:

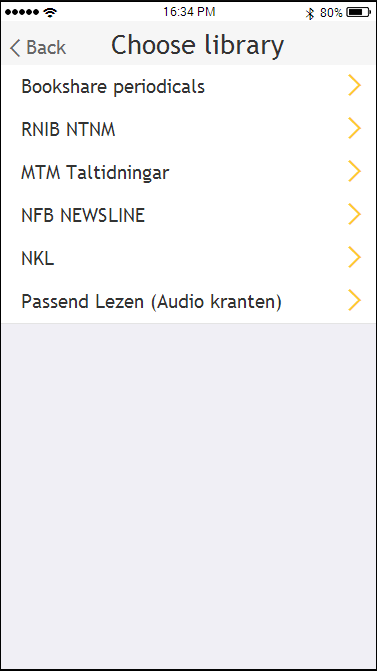


It gives the user the options to unsubscribe to the current newspaper edition (same as swipe and unsubscribe) and the option to log out the current newspaper library (e.g. “MTM Taltidningar”, “Bookshare periodicals”). After a successful unsubscribe or log out, the app needs to go back to “My Newspapers” screen and refresh the content.

NOTE: Some newspaper libraries do not have unsubscribe as an option (e.g. MTM Taltidningar) and in that case, this pop up will only show the “log out” option.

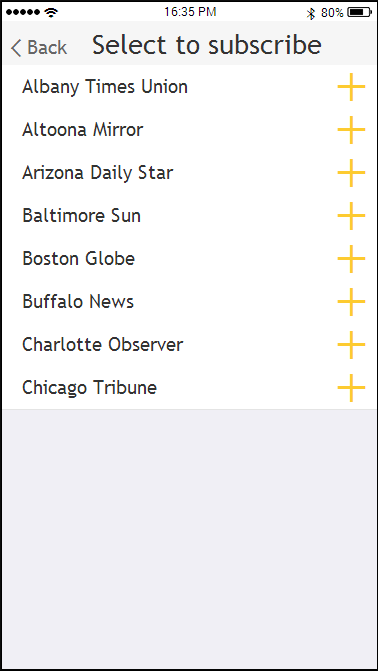
### Screen: Add subscription

When the user has selected the “plus” button in “My Newspapers”, EasyReader will display a screen where the user selects from which newspaper library to add a subscription from.



When tapping one of the newspaper libraries, EasyReader checks if the user already is logged in or not. If login is required, the login screen is displayed: [Screen: Login](#_Screen:_Login)

EasyReader now retrieves the complete list of newspapers from the selected library and displays all in a list.



Please note that already subscribed newspapers needs to be filtered out from the list before displaying on screen.

The user simply taps a newspaper to subscribe to it and this message box will be shown:



When pressing OK, it will return back to [Screen: My Newspapers](#_Screen:_My_Newspapers)

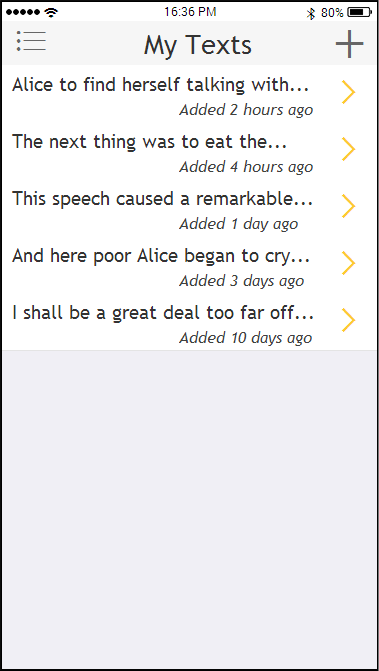
## Screen: My Texts

**User story:** As a user, I want to access the text on the device clipboard so that read the text with EasyReader’s wonderful reading features.

**User story:** As a user, I want save the clipboard text in a list so that I can access it another time and continue to read/work/study.

**User story:** As a user, I want make sure my clipboard texts are saved in the cloud so that I can continue to read the text on another device/platform.

When choosing “My Texts” in the main menu, a screen with saved texts are displayed. These texts are in a list sorted by the latest read at the top.



The toolbar at the top contains:

* Navigation button – opens the main navigation menu [Screen: Main navigation](#_Screen:_Main_navigation)
* Title label – just a large label with the text “My Texts”
* “Plus” button – pressing this button will access the device’s clipboard and check if any text is available. If so, the text will be copied and added as a new text in the list

Each item in the list represent a text that the user has added. It contains a title, which is the first x (?) number of letters in the text and information when the text was added.

When tapping on a text item in the list, the text is loaded into the reader: [Screen: Reader](#_Screen:_Reader)

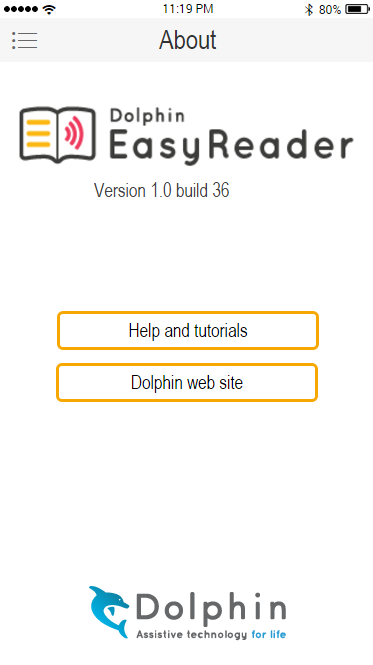
Note: when returning back from the reader, it will return back to “My Text”. Similar is for “My Books” and “My Newspapers”.

The user can delete/remove a text by sliding the item to the left and choose “Remove”.

For discussion: Should the texts be sorted by when it was added, or by latest read? This will also affect the label in each list item.

## Screen: About and help

This screen contains the about information about the app and link to the EasyReader page on the Dolphin web site.



This screen contains these controls:

* Product logo
* Text label with version and build number
* Help button – opens the device’s web browser and links to the help content (unknown URL right now)
* Web site button – opens the device’s web browser and goes to EasyReader’s product page: (<http://yourdolphin.com/easyreader>)
* Dolphin logo

## Screen: Getting Started

**User story:** As a first time user of this app, I want to get some basic help/information about the app so that can get started quickly with ease.

To be added…

# Empty screens

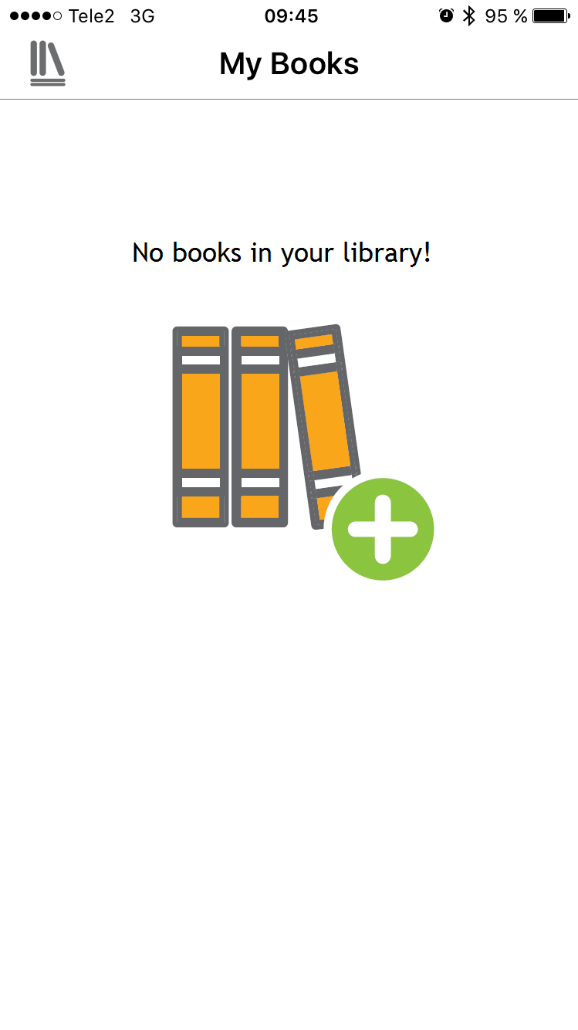
When the app is newly installed (or if the user delete all content), some screens becomes very empty and doesn’t give the user a good first impression. In some cases, the empty screen may confuse the user as they may not know what to do next. We may loose the customer at this point and to avoid this, we have decided to fill the empty screen with an image and a small text.

The empty screens is in My Books, My Newspapers and My Clipboard Text looks.

## Empty screen: My Books

If no books are available in My Books, this screen should be displayed. Compared to My Books with books, these are the differences:

* Hide the button for list/collection view
* Hide the button for sort
* Hide the search controls
* White background
* Label: “No books in your library!”
* Image: see below



## Empty screen: My Newspapers

If there is no subscriptions/newspapers, this screen should be displayed. Compared to the My Newspapers screen with newspapers in the list, these are the differences:

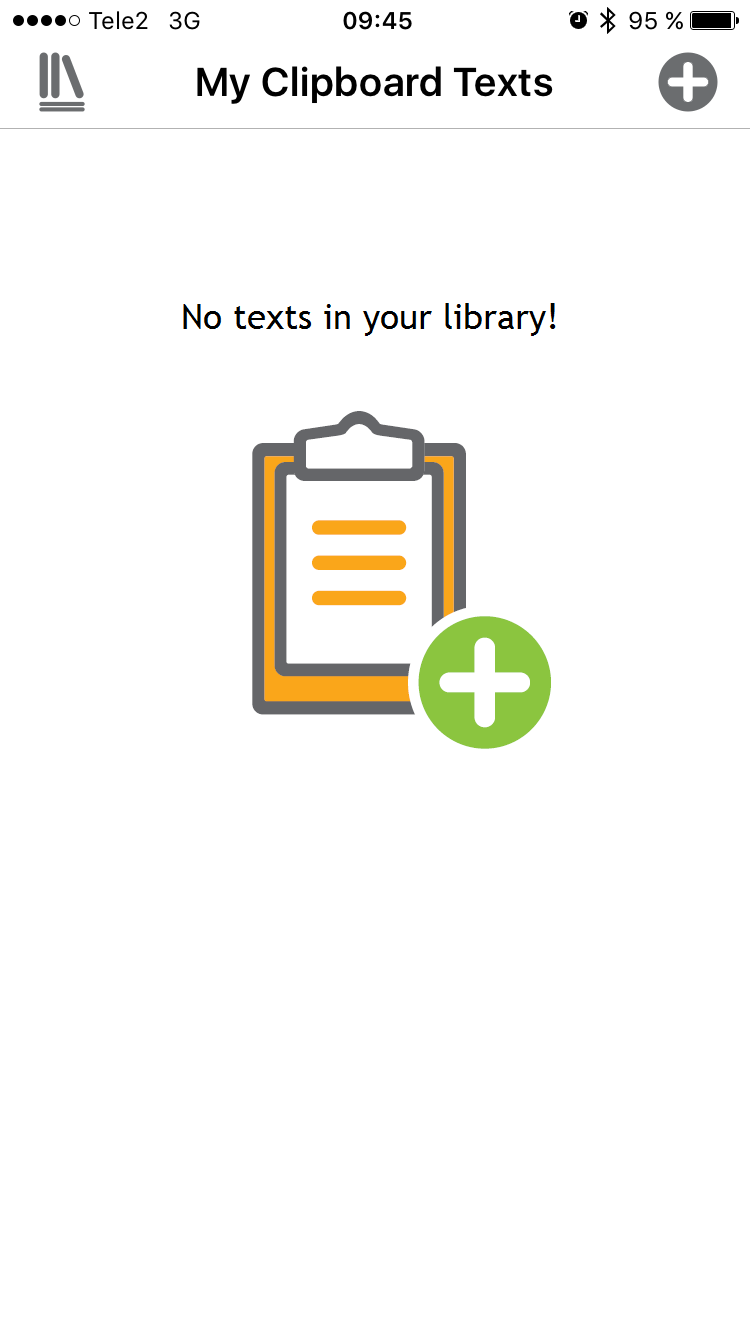
* White background
* Label: “No newspapers in your library!”
* Image: see below



## Empty screen: My Clipboard Texts

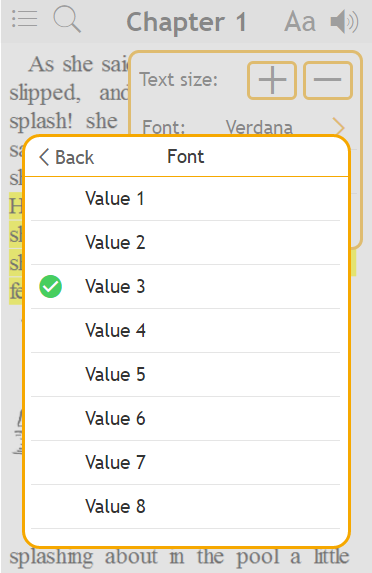
If there is no clipboard texts in the list, this screen should be displayed. Compared to the My Clipboard Texts screen with a list of texts, these are the differences:

* White background
* Label: “No texts in your library!”
* Image: see below



# Picker view window

In a few places within the app, a list of options needs to be displayed and it will use a design similar to this:



It has these controls:

* Back control in the toolbar. This may be text or graphics (to be decided later)
* Caption text, which comes from the calling control. In the above case “Font” for the font selection option
* List of available values, where the current option is preselected (if appropriate). If the number of list items are longer than the visible area, the list will scroll inside the picker view window.

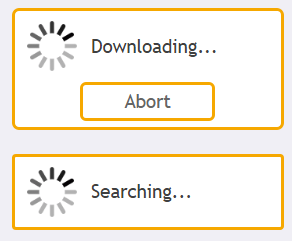
When tapping on a new value in the list, the green tick icon is moved. It doesn’t close the picker view window. When the new value is selected, in some cases the value is set directly so changes can be seen directly. E.g. the user wants to see certain text settings to be changed directly in the reader view.

When tapping “Back” or outside the picker view window, the window closes and returns back to where it came from.

This picker view window will be used for a wide different places in the app and may look a bit different depending on what it is used for. One example is the color picker view, which has a slightly different UI compared to the standard one. See last in section: [Text settings](#_Text_settings)

# Progress window

In a number of places within EasyReader, there is a need for displaying a progress window so that the user get UI feedback that something is ongoing (instead of no UI feedback at all). Below is a screen shot of two sample progress windows:



The animated progress will be delivered by the graphical designer.

The text is set by the calling process and it may say “Downloading…”, “Searching…”, “Please wait…”, “Logging in…” and so on. When there is a need for aborting or cancelling an ongoing process, this window needs to be a bit larger to fit a button.

Below is a list of places in the apps where a progress is needed:

* **Login attempt**  
  Text: “Logging in…”  
  Button: not needed
* **Entering/refresh of newspapers**Text: “Loading…”  
  Button: not needed
* **Download newspaper issue**Text: “Downloading…”  
  Button: “Abort”
* **Open book/newspaper issue**Text: “Open book…”  
  Button: not needed
* **Search text in book**Text: “Searching…”  
  Button: not needed

There might be more places in the app where this is needed, but we need to decide that case by case.

# Locked screen playback controls

**User story:** As a user, I want to have access to reading controls even if the device is locked, so that I can quickly play/stop and navigate without unlocking the screen.

When reader has a book loaded and playback is running, the lock screen on the device should display basic book information with few navigation controls.

***NOTE:*** In time of writing this section in the spec, I’m a bit unsure what the default/expected behavior should be when to display it, when not to display it and so on. May be different on iOS and Android? Investigation required.

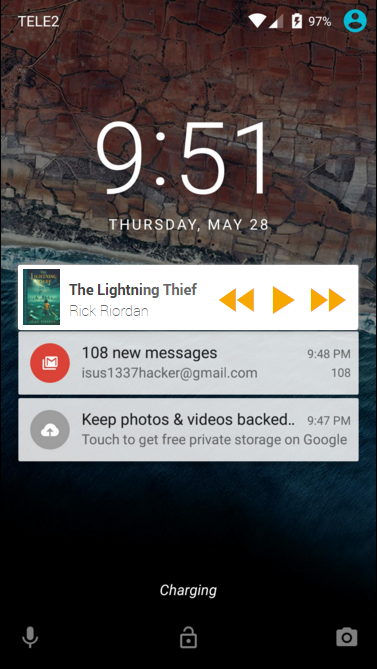
This is the iOS design:



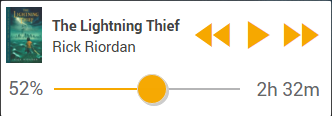
The iOS locked screen have these controls:

* Nav prev
* Play/Stop
* Nav prev
* Label with progress in percent
* Slider for percent/time navigation
* Label with current time position in book. Blank if the book has no time length.
* Label with book title
* Label with book author
* Image with book’s cover image

For Android, we will use the notification system and the locked screen will have this notification area for EasyReader:



The notification area can be expanded and will look like this:



The notification area contains these controls:

* Image with book’s cover image
* Label with book title
* Label with book author
* Nav prev
* Play/Stop
* Nav prev
* Label with progress in percent
* Slider for percent/time navigation
* Label with current time position in book. Blank if the book has no time length.

# Cooperation with other apps on the device

**User story:** As a user who downloads content in the device web browser, I want send that downloaded file to EasyReader so that I can read the text with EasyReader’s TTS voice and my preferred display settings.

**User story:** As a user with books in my Dropbox account, I want to select the book in the Dropbox app and send it to EasyReader so that I can read the text with EasyReader’s TTS voice and my preferred display settings.

To be added…

We need to make sure that other apps on the device can send files and text to EasyReader. This opens up for a much more flexible and easy to use solution when working with EasyReader and other apps. Few examples:

* Downloading an EPUB in the web browser and then send to EasyReader
* Use the Dropbox app, select a DOCX files, send to EasyReader

In all cases above, the reader module in EasyReader must open directly with the sent file. When the reading has finished and pressing the “Back” button, EasyReader will ask: “Do you want to save it to My Books?”

In addition to this above and related; read a piece of text on the web (or any app), select a portion of text, select “copy”, switch to the EasyReader app and add clipboard text into “My Texts”.

Appendix – uncomplete sections

# Dolphin Cloud

With version EasyReader 8.01 for Windows and the new EasyReader apps, we are introducing a new cloud service called “Dolphin Cloud”. The user can sign up for this service for free and Dolphin Cloud will make sure a seamless experience when moving from device to device. This is a draft idea on what EasyReader will store in the cloud:

* **The list of books in “My Books”**  
  The books on “My Books” will be synchronised across devices by storing references to the books (please note that we will NOT store any books in the cloud, as it may be problematic legally to store copyright protected content on a Dolphin server).  
  We will not store books that has been added manually from external source – only books downloaded/streamed from supported online libraries.
* **Book information**We need to store latest read position, bookmarks, etc that belongs to the book, so that the user can continue to read and use bookmarks regardless which device is in use.
* **Texts in “My Texts”**  
  The Dolphin Cloud stores all added clipboard texts that the user has added to “My Texts”. See separate section for more details.
* **My Newspaper subscriptions**Stores the information about the user’s subscriptions and which newspapers has been read.
* **Library login information (for discussion, legal?)**  
  The Dolphin Cloud securely stores the user’s library credentials so that the user is automatically logged into the libraries when moving to a new device.  
  *NOTE: We need to double check this if this is allowed or not. I suspect we can’t do this, so we need to require a login to be made on each new EasyReader device/platform.*
* **My online libraries**  
  The Dolphin Cloud stores the user’s choice of preferred online libraries, so a newly added library on one device is automatically added on the second device.
* **Application settings**Some (all?) application settings are stored in the cloud, e.g. application language and list of preferred online libraries (see above). It can also store platform/version specific data that is only applicable for a platform or version.
* **Reader settings**The Dolphin Cloud will hold reader settings such has voice name, voice speed, font name, colours and so on. It can also hold platform/version specific data.
* **Lexicons**The user’s lexicons for TTS voice pronunciation corrections will be store in the cloud so that the edits are applied across all platforms and devices.

The user can easily sign up for the Dolphin Cloud inside EasyReader and the user only need to provide a email and a password. For security reasons, we need to validate the email address through a standard email validation. The user should be up and running with this service within a minute.

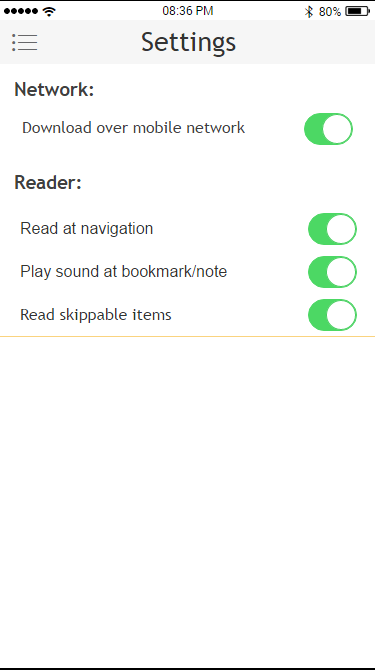
The backend system for user identification and data storage is not chosen yet, but we are considering two options right now:

* Amazon Cognito (<https://aws.amazon.com/cognito/>)
* Dolphin in-house system – we at Dolphin ran a project a while ago where we developed a web service API for Dolphin ID and data storage. That system does more or less what we need for Dolphin Cloud.

# Screen: Application settings

**User story:** As a teacher handing out tablets in the classroom, I want to be able to quickly change the application language in the app so that my students can use the app in their native language.

To be added…



In the app settings, you can access various application settings:

* **Network download settings**This is a setting that controls if book downloads are allowed in 3G/4G network, or limited to WiFi only.
* **Read at navigation**on/off switch for enable automatic “play” when doing an active navigation in the book (open book, navigate by heading, page, search result, double tap, etc)
* **Play sound at bookmark/note**  
  on/off switch for playing an audio sound when passing by a position with bookmark, note or color mark.
* **Read skippable items**on/off switch for reading “skippable” items in a book, or if the read should skip “skippable” items and continue to read after the skipped item.
* **Download location (Android only)**we need a setting where the user can decide to download books on the device or the SD card. Suggesting to use two radio button with a text above it.

To be added…

# Web browser component requirements

The apps needs a web browser component to display the book content on screen and we have a requirement of HTML5 and CSS3. Early prototype tests on both iOS and Android shows that the default web browser component on the devices have sufficient support for EasyReader’s needs. At this point, we don’t see a need for creating our own web browser component based on Web Kit, Chromium or similar. However, when we start to test on more devices, we may identify the need for this, but we will discuss that at that point.

The apps likely need a JavaScript bridge to be developed between the web component and the app code, to be able to access the web DOM and handle the communication in-between.

# Reader’s file format requirements

The reader module takes care of all different file formats that we need to support (EPUB, HTML, DAISY, etc) and provide a generic API for the UI developer to implement and support. Basically these are the different types of books:

* **Text only**This is e.g. plain text, HTML, EPUB 2 where the text is simply rendered on screen. The text may be formatted with paragraphs, list, tables, CSS-styling, etc.
* **Audio only**This is either MP3/M3U or DAISY audio-only books where there are no content to display. However, the reader module automatically generates an HTML to display something in the web browser.
* **Text and audio combined**  
  This is either EPUB 3 with Media Overlays, or DAISY text-audio books where the text and audio is synchronized – usually on sentence level.

This means that the reader need to be aware of the existence of built in audio tracks in the book and the user easily must be able to switch between audio track playback and TTS playback.

# Offline usage

To be added… Only locally available books are visible in the bookshelf (or should we visually show non-available books?). Dolphin Cloud data stored locally until Internet is connected again.

# Bar code for book search

**User story:** As a user, I want use the mobile camera to scan a book’s bar code so that I quickly can search online to see if the book is available in accessible format.

To be added…

# Streaming support of books

To be added… Limited to a DODP servers…. Default option need to be stream book, instead of download.

# Manually add books to My Books

To be added… Through iTunes, SD card, …

# Graphics required

The apps contains graphics, logos and icons that our graphical designer need to create. We need to make sure we bring the branding from EasyReader v7 for Windows into the apps, so that our users feel comfortable with the switch between devices and platforms.

Each graphical component needs to make available in different sizes, to support all screen resolutions and DPI.

To be defined later… What does iOS/Android SDKs says about this?

# Track student activities

Some user activities and user data will be stored in the cloud for multiple purposes. At some point in this project, we are planning to create a “Teacher Hub”, which is a web portal aimed for the needs at school, but also for other training organization. Part of this web portal is that the admin can go in and track the users and see progress.

This requires the user is signed on to Dolphin Cloud.

List required data to track below:

* Requirement 1
* Requirement 2
* …

To be added…

# Metrics

To be added…

Both Apple and Android provides app analytic tools which contains quite a lot of interesting metrics, trends, sales figures, territories, campaign links, web link info and much more.

In addition to these standard metrics Apple and Android provides, we are going to use metrics within the apps to measure specific areas of interest.

# App Stores descriptions

To be added…

It is very important to put quite a lot of effort in providing really good App Store descriptions for the app, as that has a direct impact on the ratio between views in App Store and number of installs.