CHALMERS


# 3Book

## - a forgiving e-reader prototype for complex books

*Bachelor thesis at Computer Science and Engineering*

Richard Davison
Axel Ljungkvist
Daniel Ström

Department of Computer Science and Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2012
Bachelor thesis nr 2012:018

**Abstract**

Navigating complex documents with heterogeneous content on a modern touch-enabled smartphone is difficult. Small screen size, limited and imprecise control opportunities and the potential of user distraction all contribute to the challenge of the task. This report describes a number of design considerations that the authors believe can be implemented to help alleviate these problems, including thumb-driven interaction and forgiving navigation. The report also describes a prototype that exemplifies the design ideas introduced.

**Acknowledgements**

Göteborg 2012-05-14

The authors would like to extend their sincere gratitude to group supervisor Kuchi V.S. Prasad for his contributions to the project. It was during group meetings that the ideas of forgiving navigation and thumb-driven design were first born. His help in discovering and exploring these ideas had a tremendously positive impact on the project.

We would also like to acknowledge the authors of the open source software that is used in the prototype:

- We use a UI control to flip between book pages that is based on the the Android Viewflow control by Patrik Åkerfeld, which is licensed under the Apache License, Version 2.0.
- We use the jsoup library developed by Jonathan Hedley to parse HTML. The jsoup library is licensed under the MIT license.
- We use the epublib library developed by Paul Siegman to unzip and interact with epub files. The epublib library is licensed under the GNU Lesser General Public License.

We are grateful that you made your work available to the public with no charge. This let us focus less on implementing things that were already out there, and more on designing the interesting stuff.

# Contents

# Definitions

**3Book** The Android prototype application based on the research and results from this report.

**Action bar** Action bar is an user interface element in android 3.0 and greater, showing the current activity's associated menu options and actions.

**Action overflow** The action overflow is a button placed to the far right on the action bar. Pressing it allows access to important actions relevant to the current activity.

**Activity** A single, focused task that the user can do. Usually provides a new view of the application, with unique controls and content.

**Android** Open source linux based operating system platform, developed by Google. Generally used on mobile devices such as smartphones and tablets.

**Control** A generic term for an interactive interface element that is used to control or interact with the software (e.g. buttons, sliders, text fields).

**EPUB** Short for "electronic publication". An open e-book standard maintained by the IDPF.

**GUI** Graphic User Interface.

**Hard-key** A hardware button, for example, menu or back key on a smartphone.

**HTML** Short for "Hyper Text Markup Language". The main markup language for web pages.

**IDPF** The International Digital Publishing Forum. Maintains the EPUB-format specification.

**Re-flow** The act of re-formatting a length of text to fit a different display size.

**Status bar** Status bar is an android user interface element showing notifications, signal strength, time etc.

**Toast notification** A notification overlay showing small white text in a black rectangle which fades out accordingly, intended not to steal too much focus from the application.

**Use case** A list of steps, typically defining interactions between a role and a system, to achieve a goal.

**W3C** World Wide Web Consortium. The main international standards organization for the World Wide Web.

# 1. Introduction

As smartphones continue to grow in power and screen size, the variety and scope of tasks that users wish to accomplish grows as well. Mobile e-book reading, once the domain of large-screen dedicated reader devices, is becoming feasible on smartphones as modern devices often feature screen sizes in excess of 3.5" (Phone-size.com, 2012). Still, the available screen real-estate is rather limited and navigation through complex documents can feel frustrating and restrictive.

If workspace area is the Achilles' heel of the smartphone e-reader, then mobility is one of its greatest advantages. While carrying only as much as one or two physical books might require a backpack, e-books enables one to carry an entire library in a pocket. This mobility, coupled with other benefits such as search and annotations, makes the option to read on your phone very desirable in many scenarios.

Currently, a number of dedicated e-book reading devices are available in the market. Such devices often provide a strong reading experience, but they are rarely inexpensive and often cumbersome. It is likely cheaper and more convenient to download an application for those that already own and carry a smartphone.

Simple text-only fiction is not overly challenging to present in a user-friendly manner even on a very small screen. Documents of greater complexity and user interaction such as textbooks are another matter; they often contain figures, tables and reference schemes. Reading such a text for its intended purpose can entail going back and forth through the document, re-reading sections, cross referencing facts and consulting tables, diagrams and indices.

Existing e-reader applications for Android often focus on bigger devices such as tablets, or fail to provide strong support for complex texts. This project explores design ideas that can be used to provide rich interaction with complex books on small screens, while remaining easy-to-use and accessible.

## 1.1 Background

**1971** The first e-book was created by Michael Hart on the 4th of July as the initiation of Project Gutenberg, the first digital "library".

**1974** The Internet is born with the introduction of TCP/IP and the first recorded use of the word "Internet" (Cerf et al., 1974).

**1991** The Sony Data Discman, an early predecessor to modern e-readers, became available for purchase (Lawinski, 2010).

**1993** A combined phone-pager-fax-pda gadget from IBM, called the Simon PDA, is released in stores. The device is even equipped with a touch-screen. (Telecommunications).

**1994** The first library website went live.

**1995** Amazon.com launched as the first online bookstore. Only the store was digital, not the books.

**1997** Many newspapers were publishing content online.

**1999** The Open Ebook format was created by the IDPF.

**2000** Project Gutenberg reached 1000 e-books and Amazon opened its digital book store with the same number of books.

**2002** The Blackberry was introduced to the market as a phone that could also be used for email and web browsing (Reed, 2010).

**2006** Google Books was launched with the goal of providing digitized and searchable books from partner libraries.

**2007** The Open Publication Structure, also known as *EPUB*, supersedes the Open eBook format.

**2007** Amazon's Kindle, one of the best known e-readers, becomes available on Amazon.com (Lawinski, 2010).

**2007** The iPhone is announced (Apple) and redefines the concept of what a smartphone is.

**2010** The iPad is released (Lawinski, 2010) along with iBooks and the iBookstore.

List 1: Unless otherwise noted, facts are gathered from Lebert (2008).

It was recognized early in the development of computers that text was a nice input format for bridging the gap between humans and computers. Consequently, keyboards and the ASCII character encoding standard were early creations in the history of computers. At the time, most computers were used as advanced programmable calculators for time-consuming numerical calculations,

but it would not be long until their aptitude for text processing was realized. The first e-book was created in 1971 when Michael Hart typed the text of the "The United States Declaration of Independence" into a mainframe at his university, storing it in a simple text file. Two years later what was arguably the first PC was released. The groundbreaking Xerox Alto was designed specifically for document processing and printing (Thacker et al., 1979).

The first e-books were mainly technical manuals and short documents for a limited audience. Reading on a computer screen at the time might not have been the most pleasant experience and the size of the computers made reading on them far less flexible than reading traditional books. The ease of distribution introduced by the wide adoption of the Internet made e-books more attractive but could not make up for the downsides. An effort to change the clumsiness of e-reading was made in 1991 with the Sony Data Discman but it was not enough to sway the masses. In 1998 NuvoMedia's Rocket eBook, a fairly capable e-reader even by the standards of today made another attempt but the market was not ready (Lawinski, 2010). E-reading only became somewhat widely adopted in 2007 with the release of the Amazon Kindle, but was not an overnight success. While the the Kindle was a capable device, it was unable to overcome perceptions of e-reading as something for "techies" or "nerds", which deterred some from investing in the product.

Since the release of the iPad in 2010, carrying a tablet is both socially accepted and even a status symbol. Tablets have quickly become a natural way for accessing web content, writing notes and checking e-mail. With the introduction of services such as iBooks and iBooks Author there are reasons to believe that the normal way of creating and consuming books could change considerably.

## 1.2 Purpose

The goal of this bachelor's project is to explore smartphone interaction design ideas for e-reading, and to develop a prototype that exemplifies these ideas. This prototype is named 3Book, and is intended to continue evolving after the completion of the project.

The long-term vision for the 3Book application is to provide a reader that meets many of the requirements of a college student reading environment while limiting the need to carry physical copies of books. This thesis aims to explain the research and reasoning behind the design principles used in the application.

The vision for the application has been stable throughout the project while the focus of the project and this thesis has been more fluid. Some time was spent researching alternate goals, which is discussed in subsection 1.2.1.

### 1.2.1 Abandoned thesis aims

The idea of supporting advanced mathematical formulae written in a structured language was brought up in early discussions with the project supervisor. Supporting formulae fit nicely into our idea of making a reader that is useful in an academic setting. The fact that the chosen format (see the format discussion in subsection 3.1.2) supports MathML made the idea even more attractive. However, it was found that creating a rendering engine for mathematics was too big of an undertaking, especially as pre-rendered images were found to be an acceptable as well as the prevalent solution for displaying math in books. Considering that MathML is a W3C recommendation it might be a worthwhile effort to implement support for the language at a later time.

Displaying musical notation was also investigated as a possible focus of the thesis. The idea was considered interesting and challenging as sheet music often run into space limitations even in print, to say nothing of the difficulty of displaying it on a small screen in a usable manner. The aim was to limit the need of bringing physical music sheets to practice sessions and study. The concept was found both unique and very engaging, but the scope would require a project of its own. As music is only tangentially related to the overall aim of the project, this idea was discarded.

In the end, while there were many interesting smartphone interaction design angles, it was considered most prudent to first work on an application prototype that could form a base for later undertakings.

## 1.3 Method

The e-reader application was partially developed under a SCRUM-like agile regimen. At the outset of development, a list of programming tasks was drawn up, estimated and prioritized. This resulted in a document which specified the programming effort required to implement the e-reader application. High-priority tasks were each assigned to a theme, and subsequently the themes were slotted into one of the available three three-week iterations that were planned.

The original plan was later found lacking in many respects as new information and knowledge was discovered. As time went by the actual work being done started to diverge sharply from the plan. A much better sense of what was required in terms of programming had been achieved midway through the project, and a new list was then drawn up. This plan was made on the basis of far more domain knowledge and was followed throughout the rest of the project.

Since there was only three project members, a rigid division of labor was never formulated, though each member focused on tasks appropriate to their specific knowledge and experience. The limited workforce and 15 week time span of the project severely restricted what was considered feasible to achieve in terms of a complete product.

### 1.3.1 Interface design methodology

The interaction design concepts presented in this thesis were developed through iterative prototyping. First the real-life situations where the application would be used was considered and the solutions used in existing applications analyzed. The analytic process can be summed up as asking four questions:

- How did other applications approach the problem?
- What are the issues with their solutions?
- How could their solutions be improved?
- What features are missing from their solutions?

Sketches of various solutions were then made, to provide a better understanding of how the solutions would look in a real application. Figure 1.1 shows an early concept sketch for the navigation scroll detailed later in this report. Unimplemented ideas like content preview and bookmarks are visible in the sketch.

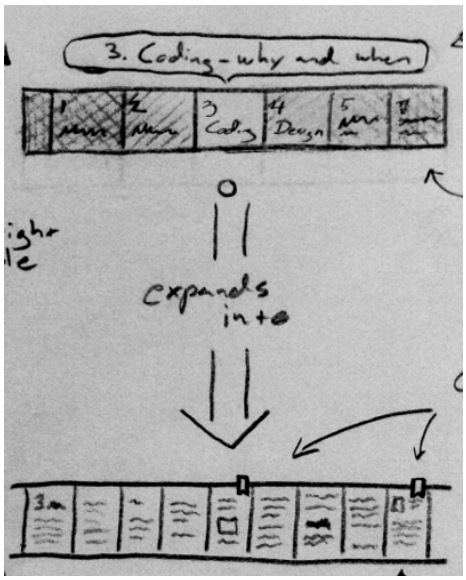



Figure 1.1: Sketch of book navigation interface.

The sketches were then discussed and improved further. Once they had reached a decent level of quality they were translated into computer graphics using an image editing program. These concepts were then given an elementary implementation in code to test their functionality on a real device. Concepts that did not function as intended or were shown to be bad design decisions were either further refined and developed or else discarded.

# 2. Design

This chapter details the design considerations that is the main focus of this project. It begins by specifying some issues with designing for reading on all small-screen touch devices and finishes by presenting some possible solutions to many of said issues.

## 2.1 The smartphone problem

The modern touch-enabled smartphone is a very young design. While laptop computers and PDAs have a history spanning a few decades, the modern touch-enabled smartphone was not seen until 2007 when the Apple iPhone was introduced to the market. This large, high-resolution touch-screen phone was the first in a class of devices that would soon become ubiquitous; companies such as HTC, Sony Ericsson, Nokia, Samsung and MicroSoft soon released devices with similar form-factors and interfaces.

Processing power has long been a concern for those developing applications for mobile devices, but in recent years the performance gap between mobile and desktop computers has shrunk. Modern smartphones can have processor speeds in excess of 1Ghz, and multiple core processors are becoming the norm in higher-end phones (Wikipedia, 2012). This means that processing power is no longer a major differentiator between mobile and desktop applications. Instead, the main challenge is now to handle the challenges and possibilities of the smartphone user interface. These challenges are primarily related to the small screen and the touch interface.

### 2.1.1 Touch interface

Touch interfaces are often cited as the most "natural" and easy to learn human-computer interface. Holzinger (2003) concurs, but cautions that they are not optimal for complex tasks. Despite this natural quality, designers of touch interfaces face a number of major constraints.

Simply translating a rich desktop UI to a smartphone is infeasible, despite the advent of handsets that have resolutions comparable to that of a modest laptop

screen. The relative imprecision of touch controls coupled with the limited physical size of the screen makes this impossible, as there is a lower limit to how small buttons can be while still remaining usable.

Park and Han (2010) found that error rates and task completion times were significantly higher for users interacting with 4mm touch keys as compared to 7 or 10 mm keys. Previous research by Parhi et al. (2006) also found that a size of at least 7.7mm was required, recommending 9.2mm or larger for all-purpose thumb use. This implies that increases in resolution will not fundamentally change the interface design considerations, as screen size then becomes the main constraint.

The issue of button size is important, as most users prefer to use their phone with only one hand(Karlson et al., 2006). The thumb is the only interaction option when the device is used in one hand. Thumb interaction makes clicks less precise and the top corners of the screen difficult to reach (Karlson et al., 2006; Park and Han, 2010). This creates tension with design conventions that mandates placing significant controls at the top of the screen, for instance the Android Design Guidelines.

Making controls too small or out of the way for thumb use limits the audience of the application, as well as the situations in which it can be used. Such designs force users to either have two hands available or else a stationary flat surface to place the device on. This can prove problematic for those who are unwilling or unable to devote both of their hands to the application.

### 2.1.2 Small screen

Smartphone screens are enormous both in terms of size and resolution when compared to older bar-style mobile phones, but pale in comparison to modern desktop monitors. While this enables designers to create interfaces for a much wider range of applications than before, the smallness of the screen is still a major constraint in the design of the interface.

Smartphone screen sizes generally range between 3 to 5 inches, with resolutions between low-end 240x320 and extremely detailed 1280x800. The original Apple iPhone is a useful mid-range benchmark both in terms of size and resolution. Its screen measures 3.5” across and provides a (since increased) resolution of 320x480 (Phone-size.com, 2012).

At such sizes it can be a challenge to fit both content and controls onto the same screen. This creates interesting problem loops when the low precision of touch interfaces are taken into account. One such problem is the balance between menu item size and number of items shown at the same time. Large items ease understanding of and interaction with the menu, but increases the risk of not being able to fit all items on the same page. The user must then navigate between several screens in order to review all menu items, which is cumbersome. However, it could be equally cumbersome to use a menu where the items have been made small enough to fit on one page, as high precision clicks are then required.

This multi-page problem is also present for non-menu content. Even modest texts cannot fit on a single smartphone screen at acceptable font sizes. The designer must thus provide some means of navigating between different parts of the content, which is often implemented by letting the user "scroll" a viewport over the content. Scrolling is an intuitive method, but several studies (e.g. Sanchez and Goolsbee (2010) and Piolat et al. (1997)) indicate that scrolling is detrimental to the understanding and future recall of the text.

It is even harder to fit both content and controls if the application deals with more than one type of content (e.g. text and images), as different types of media require different modes of interaction. Modern e-books are usually text-focused but can contain images, tables, video and audio in addition to the main theme. It is difficult to fit the various controls needed into an application that should be able to handle all such media on a small screen.

## 2.2 Solving the problem by design

The solution to the smartphone problem is good interface design. This section presents our solutions to the problems outlined in the previous section.

### 2.2.1 Thumb-driven interface

As noted in subsection 2.1.1, users often prefer to interact with their devices using a single hand and its thumb. Ensuring that all of the functionality is usable with a single thumb, provides benefits beyond catering to the preferences of the users. It opens up new usage scenarios for situations when the other hand is otherwise occupied, and creates a level playing field for those with medical conditions that preclude the use of two hands.

Ensuring thumb-only access can be seen as a mobile variant of the "Keyboard Only" pattern suggested by Tidwell (2005). It is important to note that designing for the thumb does not imply that multi-touch gestures are prohibited; rather, it means that there should always be a thumb-only alternative available. One such example is providing zoom in/out buttons for image views, while still retaining the standard pinch-to-zoom gesture in the view.

Designers concerned about cluttering their interfaces with "unnecessary" controls could provide a menu option that hides the thumb-helping interface elements. This allows the users to make choices about which interface is best for them, instead of prescribing a solution that might not be a good fit for everyone.

### 2.2.2 Forgiving navigation

Forgiving navigation is a design concept that was introduced to deal with low-precision touch controls and the cramped nature of the small screen. It is all too easy for the user to make mistakes even with generous control sizes. Designers

should pay attention to aspects of the interface that demand high precision from the user as well as interactions that exact a high price when a mistake is made.

The user is not at fault for making click mistakes or misunderstanding the purpose of the interface. Smartphones are a fairly new class of devices with a novel interaction model. Over 850000 devices are activated each day in the Android ecosystem alone (Rubin, 2012), which means that a lot of users are experiencing touch interaction for the first time. Smartphones are also frequently used "on-the-go" or in situations when the user cannot fully concentrate on their interaction with the device. Such circumstances makes users more error prone, meaning they could benefit from forgiving designs.

### 2.2.3 Content-specific views

The lack of screen real estate makes it difficult to place content and controls on the same screen. A single modern e-book can contain several different types of content such as text, images, tables, and video. The user interacts with these diverse kinds of contents in different ways, and needs unique controls for each. This is difficult to provide using only touch input, as the small screen size restricts how many interface elements can be placed.

By shifting each type of content to its own specific view both screen and gesture space is freed up. This allows rich interaction tailored to the specific content that is shown. Video and audio content can use on-screen swipes to fast-forward or rewind, while images and tables can be panned with the same action. Placing thumb-friendly interface elements is easier, as screen space is freed up by removing controls for other types of content.

### 2.2.4 Advanced navigation

Textbooks are increasingly often available as e-books, and while the e-readers of today are fairly good at displaying novels, textbooks require a device with different capabilities. Reading a textbook is much more of a dynamic exercise than a linear progression. Often different sections of the book is read in parallel or cross-referenced. Footnotes and references introduces the need to jump to a certain point in the text and then quickly return. Certain parts of the book might be desirable to quickly jump to at any point in reading. These examples highlight the need for excellent and multi-faceted navigation options.

Textbooks are often long and well-structured, with several hierarchical levels of structure. This structure should be made easily accessible to the reader in such a way as to allow the reader to get a clear overview of the anatomy of the book. Such an overview permits quickly moving about in the text and supports a dynamic way of reading. Efforts should be made to minimize the amount of loading required to switch between different parts of the book with the goal of removing any time penalties.

A crucial part of the navigation design consists of the proper use of the back-

button. After having looked up a reference or cross-referenced a different chapter, an easy method of return should be provided to the user. Keeping a history of navigations in memory, a simple touch of the back hardware key should bring the reader to the original position in the text. Only major navigation actions, such as jumping directly to a chapter or following a hyperlink, should be remembered. This allows a user to jump to a position in the vicinity of the target section, explore the surrounding content using the standard flip-page forward/backward gestures, and when satisfied return to the original position.

In a scholarly context, one book is seldom used on its own. A textbook about physics might be used in conjunction with a reference book. In such a case, easy and quick switching between the different books is of vital importance. Each book should have its own navigation history and switching between several open books should be quick and effortless.

# 3. Implementation details

A large portion of the time invested in this project has been put towards creating a prototype application as a "proof of concept" of our interaction design ideas. This chapter describes implementation details that were critical to the creation of the prototype.

## 3.1 Format considerations

One of the main arguments for mobile e-book readers is the convenience inherent in being able to carry an entire library in one's pocket. This vision of "one reader, infinite books" is severely compromised by the myriad of e-book formats that are in use[1]. This fragmentation is caused by publishers and e-book vendors who often only make their texts available in a small range of formats or even a single format, which means that the avid e-book user must keep a stable of several reading devices and applications. A strong e-book reader application should provide support for as many formats as possible in order to provide greater user convenience.

Supporting multiple formats in order to present a unified reading experience is a goal of the final product, but not necessarily of the prototype. For this reason the code behind the prototype is focused on handling the single format, but designed for future format extensions. This approach means that all formats that provide a minimum degree of functionality, mainly related to structure, can be implemented.

### 3.1.1 Format requirements and design space

In choosing a file format one must consider more than simple ease of implementation. The choice of file format strongly constrains the design space available as different file formats provide different feature sets. This means that the choice of format used in the application must be based on the requirements derived from the design concepts.

Making a strict definition of a page size or shape was considered detrimental

[1] No less than 17 e-book formats are listed on Wikipedia at the time of writing.

to the reading experience, due to the varied screen sizes of Android devices. Therefore it was decided that the chosen format had to be structured without the notion of pages, so that its contents could be "re-flown" for each screen.

Presenting the information in such a manner, appropriate to each specific device, requires that the format provides a facility for separating styling from content. The presentation is further enhanced when the format exposes semantic information, as it is then easy to differentiate between distinct elements such as headings, images and body text.

Another major requirement was access to the structure of the text. This would support quick navigation between different parts of the document. Ideally, the format should describe several levels of structure so that chapters, sections, sections etc. are all accessible in an easy manner. There was also a desire to transcend the limitations of the physical book by using the full capabilities of the smartphone. Alongside text and images, an author should be able to include audio and video content. This could be especially useful for educational textbooks.

Finally, the ideal format should be an open industry standard, utilizing technologies that are simple and well-studied. Formats based on technologies that the project members had experience with were also preferred, given the narrow time constraints of the project.

### 3.1.2 Results of format pre-study

A survey was done of the popular e-book formats where the features of the formats were assessed in relation to the requirements discussed above. EPUB was consequently chosen as the format supported by the prototype, as it met all of the requirements. It is an open standard maintained by the International Digital Publishing Forum (IDPF) and based on standard web technologies such as HTML and CSS. This means a clear division between content and styling is present in the format. The latest version of EPUB (EPUB 3) also makes use of new HTML5 and CSS3 features to support the embedding of video and audio, among other advanced features.

The forgiving navigation concept presented in subsection 2.2.2 carries close ties to ease of use and accessibility and EPUB 3 supports many accessibility features through the integration of the Digital Accessible Information SYstem (DAISY). Finally, EPUB is currently the most widely accepted industry standard among XML-based formats (as opposed to e.g. PDF). Today, EPUB-books are offered by Project Gutenberg, Google Books, iBookstore, Barnes & Noble and Sony to name a few, and is supported by most notable e-reading devices.

## 3.2 HTML Renderer

As mentioned in the prior section, EPUB consists primarily of of HTML. At first it was believed that this would save a lot of implementation time due to the availability of a pre-implemented WebKit-based[2] HTML viewer called WebView. Instead, halfway through the project it was found that some features would not be feasible to implement using WebView.

### 3.2.1 Webview issues with pagination

The design of the application calls for presenting text in paginated form rather than allowing users to scroll the text. Pagination requires less interaction with the application to read a given length of text. This allows readers to focus on reading rather than interacting with the application.

In order to divide content into pages the application has to take the size and resolution of the reading area into account, filling it with words until full. Doing this in a WebView requires sending a large amount of data between a source-controlling Java side and the browser-interacting JavaScript side. JavaScript is about three times slower than Java in an Android environment (Pala, 2012). This performance loss might have been bearable, but the busy interconnect between the two systems was empirically found to be a massive bottleneck that brought the responsiveness of the application down to unacceptable levels.

### 3.2.2 Webview issues with dedicated views

One of the main features of our application is dedicated and optimized views for different types of content (see subsection 2.2.3), such as images and tables. Implementing such views using WebView would require a large amount of Java to JavaScript communication, which was already found to be unbearably slow.

### 3.2.3 Implementing a partial HTML-renderer

Since WebView did not meet the requirements of the application, another solution had to be found. As no existing component was sufficient, a decision was made to implement a new rendering engine that would solve these issues. This new engine was to provide full access to every aspect of the contents of the book.

A complete HTML parse library called jsoup was used to provide programmatic access to the HTML tree structure. This tree was then converted to a linear list form using a standard tree traversal algorithm.

The linearized HTML is rendered to a canvas on a page by page basis. Rendering text to a canvas requires rather low-level programming but in return provides

[2]WebKit is the rendering engine used in the Chrome and Safari web browsers.

full control over what's rendered in terms of color, typeface, font-size, font-spacing, line-height and font-style. This control makes it possible to provide the users with detailed options to change how the text is styled and presented.

Rendered pages are stored in a slot buffer that keeps a render of the current page and two pages before and after the current one. This reduces perceived render times unless the user is paging very quickly. The renderer renders a regular page on a modern smartphone in about 50 milliseconds.

It should be noted that the renderer cannot render every tag in the HTML specification. It has support for the tags most commonly used in EPUB files, and is intended to evolve and grow as needed.

# 4. Result

This chapter presents the prototype that was developed to determine whether the proposed designs were feasible to implement. It also presents some designs that could not be implemented due to time constraints.

## 4.1 The prototype

This section contains screenshots and descriptions of the most important components and screens of the prototype. The underlying interaction design principles are described in the second part of the design chapter, and exemplified at the end of this chapter.

### 4.1.1 Start screen

The start screen is designed with a "less is more" approach, presenting the user with a minimalistic screen on application start as seen in Figure 4.1.

Only four menu options - "My collection", "Favourites", "Open/Import" and "Settings" - are available on the start screen. This makes it easy for the user to see which actions are available. The buttons, divided by a grid, cover the remaining area of the screen. This makes them very easy to target, minimizing the risk of unintended navigation.

The action bar on this screen only contains a search action which searches through the collection. The search query can be a specific book title, a tag or an author.

Below the start screen action bar is a horizontally scrolling list view of the users recently opened books, arranged in chronological order. This provides quick and easy access to the most recent books, without forcing the user to dig through menus or browse their devices' sd-card.

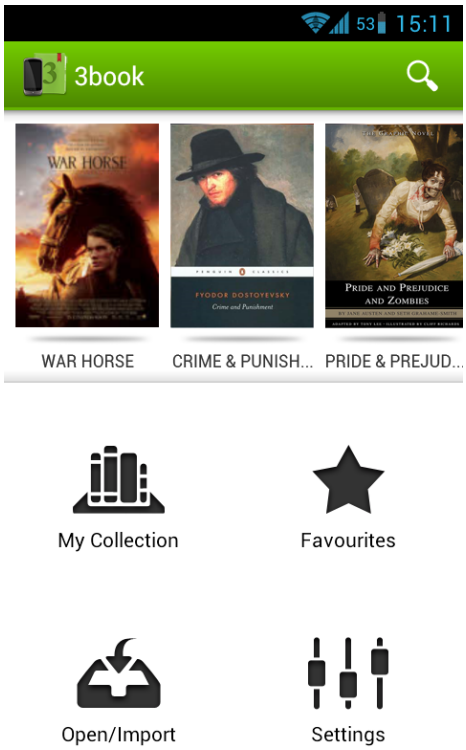



Figure 4.1: Start screen.

### 4.1.2 My collection

The "My Collection" activity presents the contents of the collection in three different view fragments (see Figure 4.2). A fragment is a reusable component or behavioural specification of an Activity. The views are Books, Authors and Tags. The book tab shows all books, the authors tab filters books on a per-author basis, and the tags tab filters books based on user-provided tags.

The user navigates between the tabs either by a horizontal swipe gesture or by touching the corresponding tab. The application remembers which fragment was last visited and displays it when the activity is relaunched.

This screen uses a split action bar. The main action bar provides quick access to search and settings, while the bottom bar allows the user to filter and reorder content depending on the current fragment.

Each fragment contains a list view of books, authors or tags. The items in the book fragment can be easily distinguished due to the different book cover art on each row. A small progress bar is shown on top of the cover art, giving the user visual feedback on where they stopped when the book was last read. To allow cutting through menu hierarchies, an action icon is placed to the left on each row. This action icon allows users to invoke special rarely used actions such as

"delete" and "details".

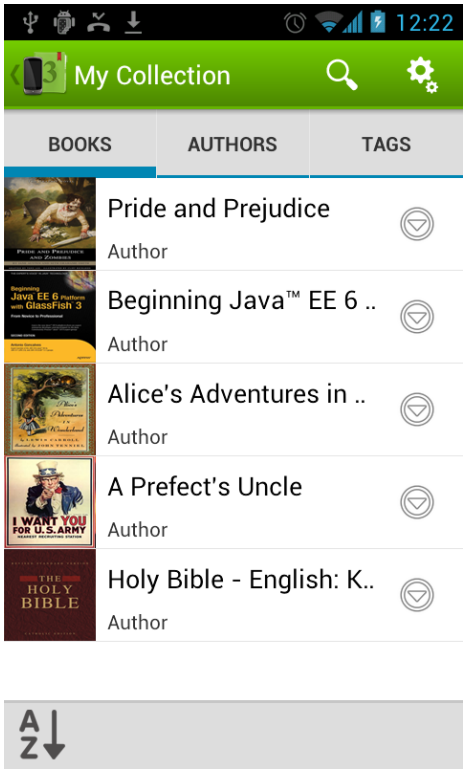


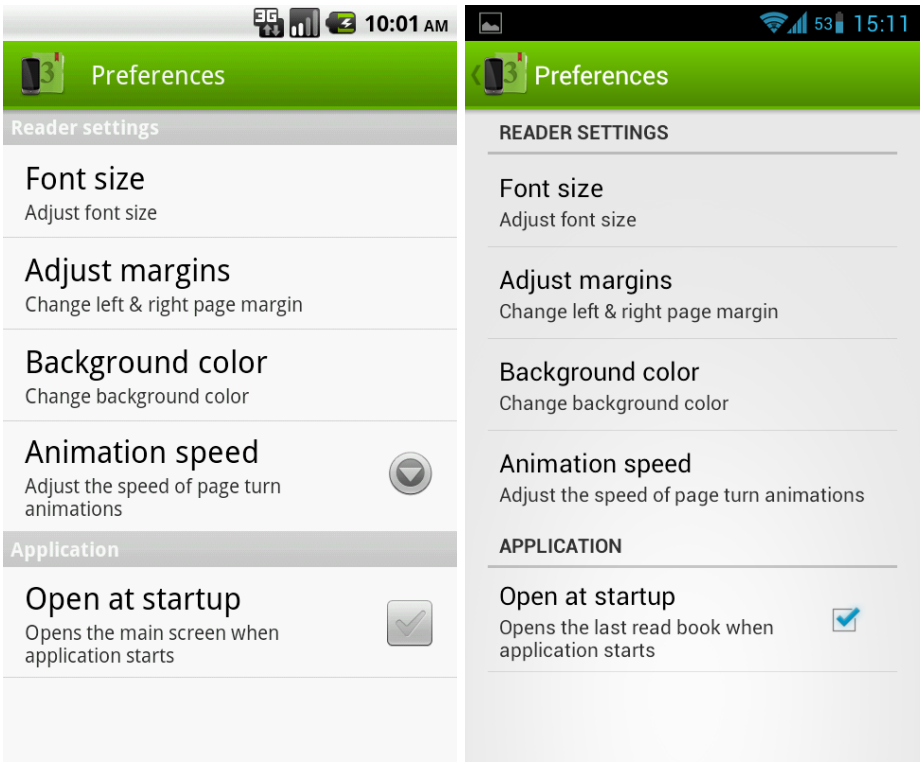

Figure 4.2: Book collection screen.

### 4.1.3 Settings

This screen presents basic user-adjustable settings, divided into sections to provide a better overview. As shown in Figure 4.3 the appearance of the application adapts according to the operating system version in order to achieve consistency and familiarity.

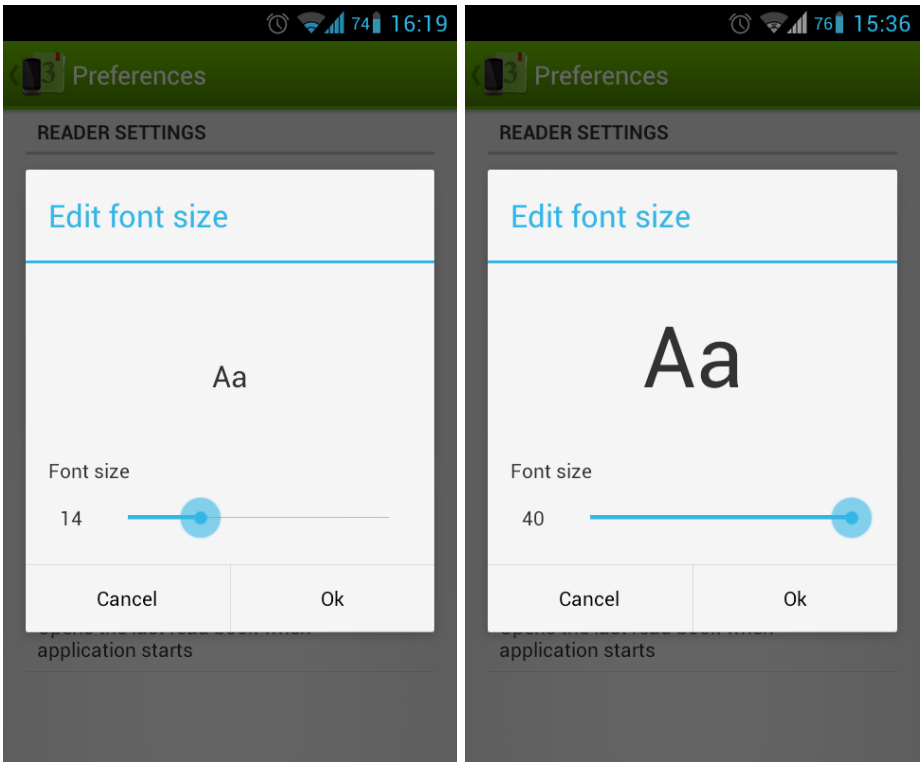
Settings influencing how the text should be rendered (such as margins and font-size) are provided with a visual feedback element. The visual feedback element is positioned so that thumb occlusion is minimized (see Figure 4.4). The most important of these settings are also accessible from the reader screen, saving users a few navigational steps.

The action bar on this screen provides no other actions then the up-action.




(a) Android 3.2 Gingerbread. (b) Android 4.0 Ice Cream Sandwich.

Figure 4.3: Settings screen on various platform versions.




(a) Small font size.

(b) Large font size.

Figure 4.4: Visual feedback for font size setting.

### 4.1.4 Open/Import (file-browser)

The file-browser activity (see Figure 4.5) is launched when a user touches the Open/Import button on the start screen. This screen contains a listview populated with the contents of the devices' storage. Icons are used to distinguish directories from files and various file-types. Files that can be read by the application are represented by a special icon. The list is populated first with directories and then with files, with both sections sorted alphabetically. This makes it easier for users to quickly identify their files and e-books.

The contents of the sd-card is displayed at startup, as this is the default user-accessible root directory. Touching one of the list view items invokes different actions depending on file-type:

**Directory** The contents of the directory are displayed, taking the user deeper into the file tree.

**Unsupported File** Nothing happens. This list item is disabled and no visual touch feedback is provided in order to demonstrate that this is an unsupported file.

**Supported File** The book represented by the file is opened if the file is already present in the collection of the user. Otherwise, a dialogue is displayed. The user can choose between opening the book immediately, importing the book into the collection, or aborting the action. If import is chosen the book is subsequently opened.

If a row in the displayed list represents a directory or a supported file, a checkbox is aligned to the right of the row. A contextual action bar that allows users to import multiple files is displayed when at least one checkbox is checked. The checked subdirectories are recursively scanned for e-books when the action is invoked. All import actions are cancelable, and they display a progress bar which allows users to determine progress.

The bottom of the screen contains a bar displaying the current file path. The main action bar on this screen has a parent directory action, which allows users to change the directory displayed to the parent of the current directory. The "back" button functionality is overridden in this activity. Here it invokes the parent directory action, unless the current directory is the root directory.

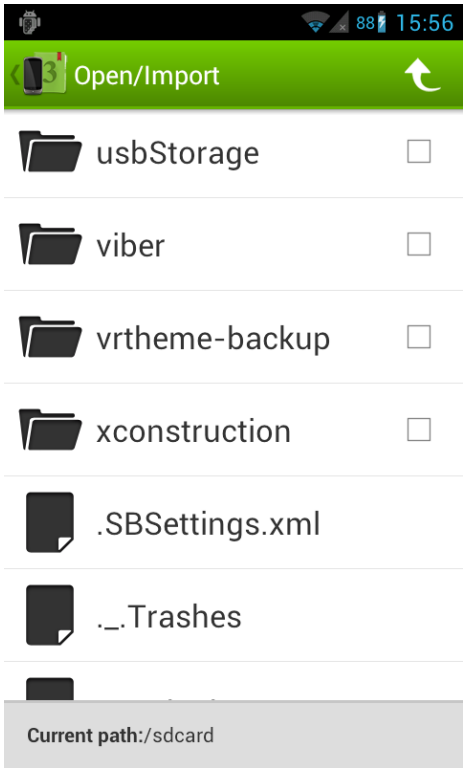
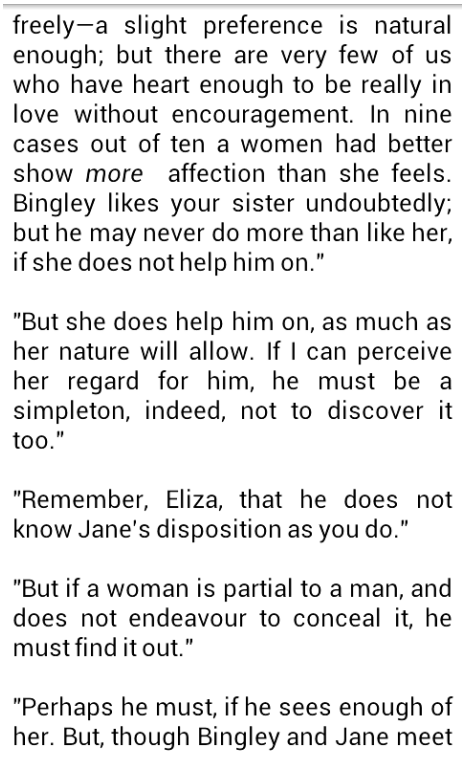



Figure 4.5: File browser screen.

### 4.1.5 Read screen

The read activity is the most important screen of the application, as this is where the users read their e-books. Users are presented with a full-screen representation of a book page when the activity is launched. The first page of the book is displayed if it's the first time the book is read, otherwise the last seen page is displayed.

The page is actually a rendered bitmap of the current content in the e-book file (See Implementation details). As Figure 4.6 shows, no action bar is displayed and the Android system notification bar is hidden. The bar is hidden to provide less distraction from the reading experience and to give more room for content, which according to Tidwell (2005) helps create a sense of flow.


freely—a slight preference is natural enough; but there are very few of us who have heart enough to be really in love without encouragement. In nine cases out of ten a women had better show *more* affection than she feels. Bingley likes your sister undoubtedly; but he may never do more than like her, if she does not help him on."

"But she does help him on, as much as her nature will allow. If I can perceive her regard for him, he must be a simpleton, indeed, not to discover it too."

"Remember, Eliza, that he does not know Jane's disposition as you do."

"But if a woman is partial to a man, and does not endeavour to conceal it, he must find it out."

"Perhaps he must, if he sees enough of her. But, though Bingley and Jane meet

Figure 4.6: Read screen.

To navigate within this screen and invoke different actions, some basic gestures are used. These gestures all practise our "forgiving navigation" concept:

**Single tap in center of the screen** Displays the "read overlay" view, the system notification bar and the action bar. The center of the screen is defined as the vertical box that is in the center-most 33% of the horizontal space available.

**Swipe left/right** Changes the page. The sensitivity of the gesture depends on how fast you are swiping; slow swipes require you to swipe over at least 50% of the width of the screen, in order prevent unintentional page changes. Left and right sides of the screen are defined by the vertical box that is in the right- or left-most 33% of the horizontal space available. The vertical accuracy of the swipe is unimportant as single-finger vertical gestures have intentionally been left unused.

**Tap on left/right side of the screen** Changes the page. Users are required to touch and release in the same area to prevent unintentional page changes.

It is also possible to use the hardware volume keys to change pages; volume up switches to the next page while volume down switches to the previous page.

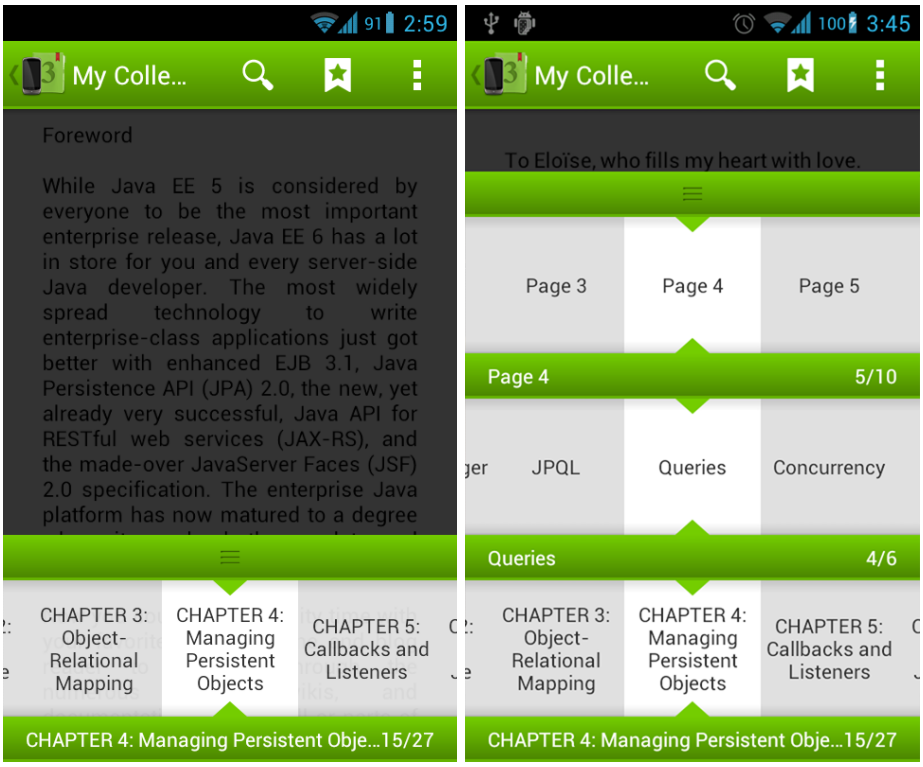
### 4.1.6 Reading overlay

The read overlay has a black background with 30% transparency to help users feel that they are still on the same page in the book. This view also features a *chapter scroller* on the bottom of the screen. This navigational scroll view allows users to easily scroll into different hierarchical sections of the book.

Figure 4.7(a) shows the scroll view in its collapsed state. Dragging the handlebar creates new instances of the scroll view underneath the top level scroller. These new scrollers are are populated with items based on the structural children of the selected item in the bottom-most scrollview. For example, dragging the handlebar of a chapter will produce a scroller with the sections making up that chapter. Further dragging of a section displays subsections (if available), and so on. The first page of the chapter, section or page is shown when the corresponding box is clicked.

Figure 4.7(b) shows the controls that allow the user to navigate between chapters, parts within the selected chapter and pages within the selected part on the same screen.

The multi-level structure is designed to make it easy to drill down to a specific part of the document. It enables the user to quickly go back and forth between different sections of the book once the relative distance between the sections of interest is known, as a single fling can scroll between 10 and 60 chapters in a very short time. If the user over- or undershoots the target it's easy to correct the mistake with smaller swipes.




(a) Collapsed navigation scroll view.

(b) Expanded navigation scroll view.

Figure 4.7: Overlay view.

The action bar contains the bookmark and search functions when the overlay is open. Choosing which actions to expose here is an important decision, as this action bar likely is the one that will see the most use. These two actions were chosen over other important functionality such as text-resizing and day/night-mode, based on the expectation that users would more often like to use search and bookmarking. Users can still access such functionality through the action overflow icon.

### 4.1.7 Dedicated views

Dedicated views for non-text content is an important feature of the application. Due to time constraints, only the image view was implemented. While fairly simple, it is still a good demonstration of the concept.

Placing the image in its own view rather than as a part of the text view allows users to see the entire image at their magnification of choice. This makes complex diagrams or large illustrations usable for mobile users.

The design of this image view also exemplifies part of the forgiving navigation concept. It provides a thumb-friendly alternative to the common multi-

touch gesture “pinch-to-zoom”. A sliding zoom control is displayed when a user touches the images, and fades away after use. Multi-touch is still available, but the redundant controls give the users greater flexibility without being overly distracting. This design is fully inclusive of those who only use one thumb.

Images that fit entirely inside the reading area do not need a special view and are simply shown as-is. Larger images are provided with a wrapper around a cropped thumbnail of the image. This wrapper, as shown in Figure 4.8 has a distinct trigger button. This helps differentiate between the two cases, and also aids user discovery of the full-screen functionality.

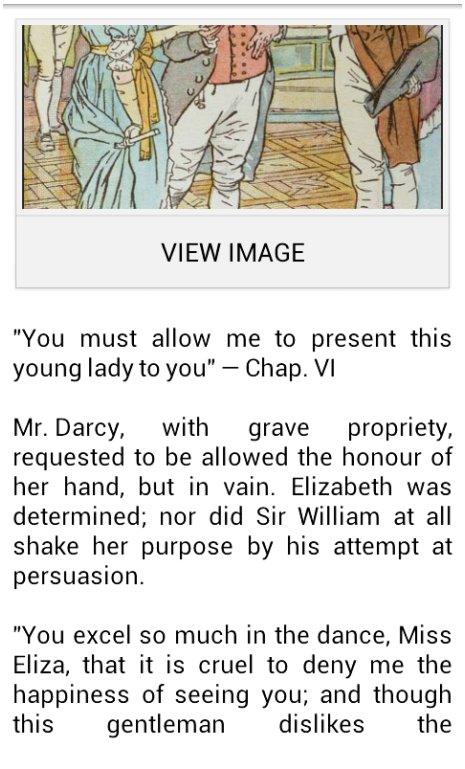



Figure 4.8: Image object with clickable wrapper.

Figure 4.9: Image view with thumb-friendly zoom control visible.

## 4.2 Un-implemented designs

This section covers some design concepts which were left unimplemented due to time constrains.

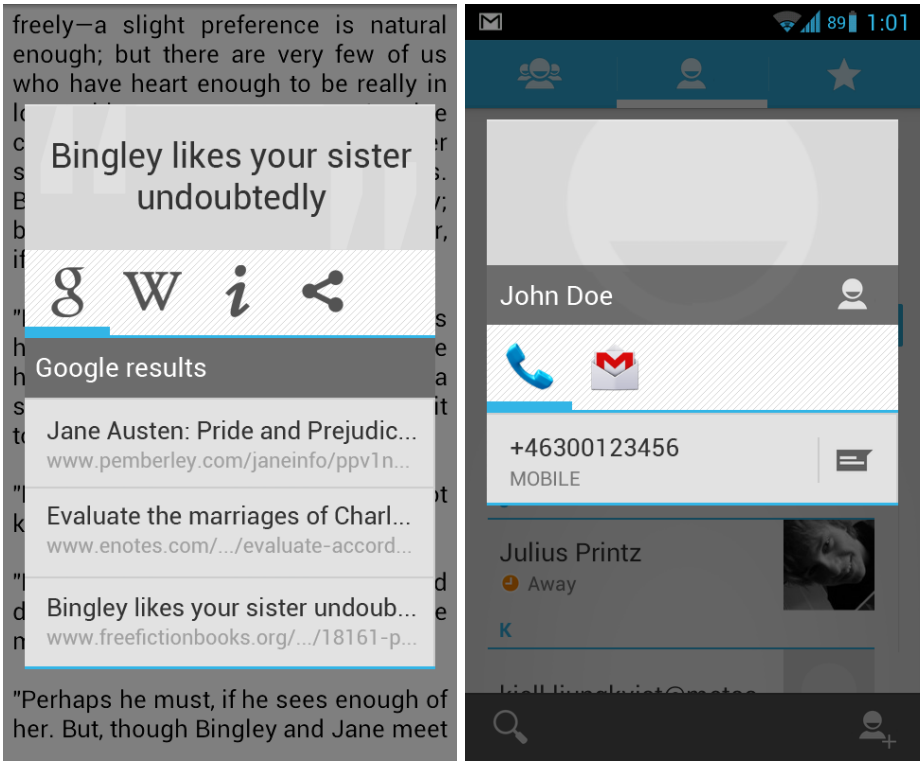
### 4.2.1 Text select dialogue

One unimplemented design is the interactive text select dialogue shown in 4.10(a). Users can select text by performing a long-press on any text element in the read activity. The pressed word changes background color to indicate its selected state and two draggable "handlebars" are displayed to the left and right of the selected word. These bars can be dragged to change what is selected. Once the user taps the range of text the text select dialogue is displayed.

This dialogue mimics the design of the convenient "contact card dialogue" known in Android 4.0 but provides different functionality. Different tabs can be selected for various features as in the contact card dialogue. The figure illustrates real-time search results from Google since the "G" tab is selected. Wikipedia results and word definitions are similarly displayed while the "share" tab provides sharing options. A tab label is also provided since the icon metaphors

are not completely self-explanatory and could have multiple meanings.

The dialogue provides shortcuts to actions that would otherwise require the user to launch several other applications. The dialogue thereby helps the users to maintain focus on reading since they stay within the application. For example, if a user touches a Google result from the list, the browser immediately displays that result and a simple back press takes the user back to the reading application.




(a) 3Book text interaction dialogue.

(b) Android 4.0 contact card.

Figure 4.10: Interactive features with a selected text compared with legacy contact card. From left to right: Google search, wikipedia search, in-book search and sharing options.

### 4.2.2 Dedicated table view

Figure 4.11 illustrates another unimplemented design. Like the dedicated image view, this view is optimized for one type of content: tabular data.

Tables are shown in the reading view as a thumbnail surrounded by a wrapper with a distinct "view" button, similar to how large images are presented. Touching the button launches the table view, which displays the data in a zoomed out state, as shown in 4.11(b).

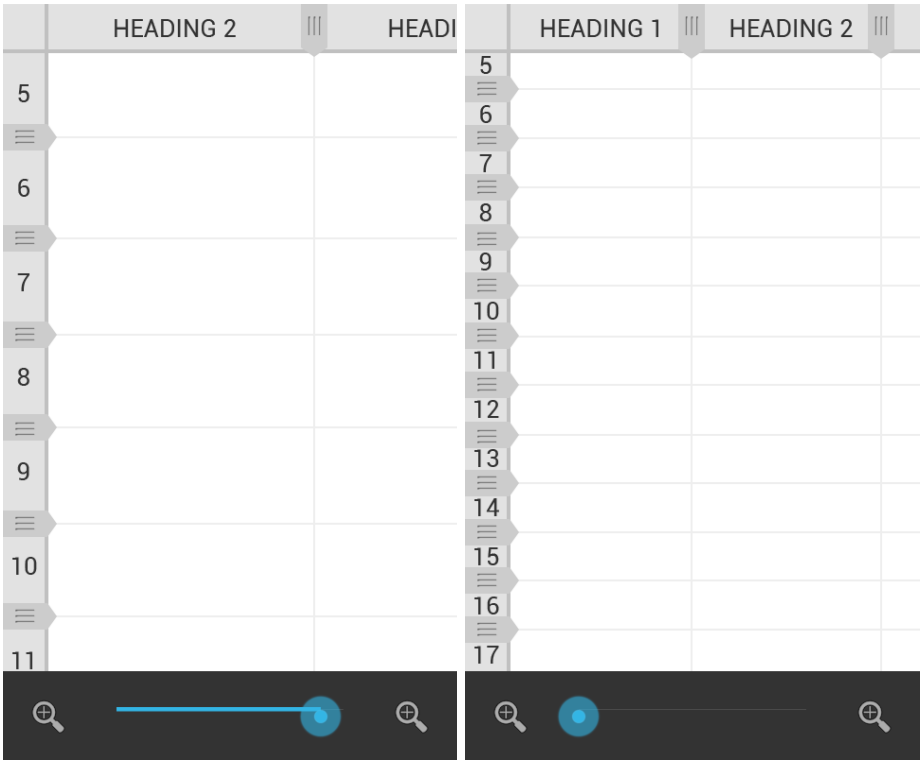
Zooming the data can be done with a pinch-to-zoom gesture or by using the

zoom controls, which are located in the bottom of the screen to maintain consistency with the rest of the application. Each individual row and column are resizable via the "handlebars" that are set in the space between two such elements. Double-tapping any header item or row number adjusts the cell size to fit the content entirely. A swiping gesture over the cells will pan the data viewport while header items and row numbers remain locked in position.

Rows and columns can be long-pressed to bring up a menu which offers the user the option of moving or hiding the row or column. A "hidden" row or column still takes up a few pixels of real estate which the user can zoom in and long press to display a menu option to restore the column to its previous size.

Tapping a column header causes the application to attempt to sort the table in ascending order based on the content of that column. A second tap reverses the sort order while the third tap brings the table back to its unsorted state.

This specialized table view provides more interactivity compared to a standard HTML table. The resizing, moving and hiding options let the user configure the table to focus on the parts of the table which interest them. These options, together with the zoom feature, also help reduce the need for horizontal scrolling, which has been found to greatly increase user seek time and reduce usability (Kim and Albers, 2003).




(a) Zoomed in table. (b) Zoomed out table.

Figure 4.11: Tabel data in dedicated view.

## 4.3 Forgiving navigation examples

One of the chief principles developed during the project has been the notion of forgiving navigation discussed in subsection 2.2.2. Being an extension of Tidwell's (2005) "safe exploration", forgiving navigation can and has been applied in all parts of the application.

One example of how forgiving navigation has been considered in 3Book is how progressing through the book is done. On the read screen, users are limited to progressing by pagefulls of content through touching the edge of the screen or swiping across it. Scrolling, such as when reading a web page, is disabled since this interaction promotes frequent or continuous scrolling while reading. Instead the application itself determines how much content fits on the screen presents this, requiring the least amount of cognitive effort from the reader. This makes for a transient page-flipping experience which helps the reader to focus on the content.

Disabling vertical scrolling also hold the merit that vertical movements during swiping can be ignored. This means a swipe doesn't have to be perfectly horizontal to work and making page-flipping less error prone. In contrast, the popular book-reading application Moonreader+ supports both paging and scrolling of content, but an inexact page-flip swipe can be interpreted as a page scroll action, with unexpected results.

Another forgiving navigation method is the chapter scroller described in the section "Reading overlay". The user is free to browse through the different chapters without any change to the current reading position. If the correct chapter or section is found a jump to the new position is achieved through clicking it's box in the scroller.

The boxes representing a position within the book are all very large (each side is longer than 15 mm), which makes misclicks unlikely (Park and Han, 2010; Parhi et al., 2006). Misclicks that still occur are easily corrected, as very few items are shown at a time. The user can simply bring up the scroller with another tap and re-select the correct item, which is likely adjacent.

In order to expand a chapter into a list of sections the chapter's box has to be at the center of the scroller. Positioning the box correctly can be finicky when done manually, so the center-most button is automatically positioned in the middle of the scroller once the user stops touching the screen. The user can then either click the box to display the chapter, or drag the handlebar upwards in order to explore the sections of the chapter. The same goes for the expanded scroller, where clicking takes the user to the section and dragging exposes sub-sections or pages.

The greatest downside of the design of the chapter scroller is the rather low information density, which makes it more difficult to get a wide overview of the breadth of content at a certain level. This is a problem for users that are not acquainted with the overall structure of the book. For this reason, a traditional "top-down" table of contents is available on a separate screen, which presents a wide, rather than a deep, view of the contents.

In contrast, Aldiko (a popular e-book reader), uses a "seekbar" component to navigate within a book (See Figure 4.12). This works rather well for short books, but a large number of pages requires extreme precision on the part of the user, as only a few pixels of movement could result in jumping an entire chapter.

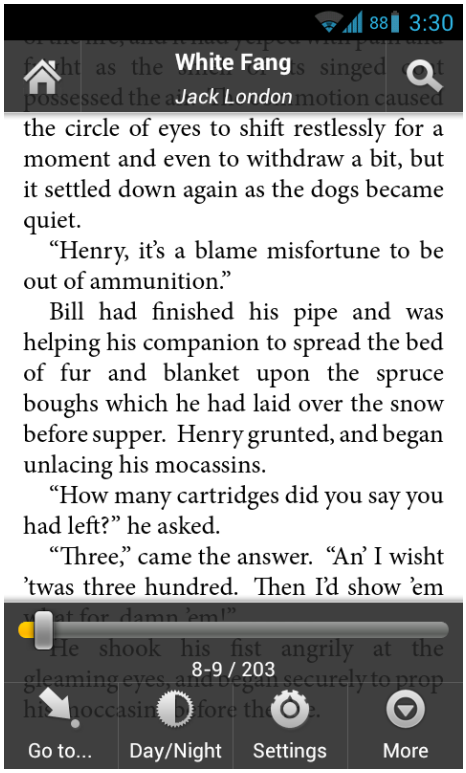



Figure 4.12: Aldiko book reader application showing overlay.

# 5. Discussion

The final designs and principles are discussed at length in chapter 4 and section 2.2 but many times in choosing between solutions, several good points has been made both for and against the final choice. This chapter discusses some of the design choices made, the current state of the prototype and highlight some examples of how design principles has been worked with in the interface.

## 5.1 Vendor guidelines vs usability

One major discussion during the course of the project has been whether to adhere to the Android design guidelines or design the application based entirely on our own research and ideas. Android is an operating system used on many different devices and as such it is difficult to create a standard that fits every single device perfectly. Several guidelines prescribe design-patterns that seem to favor two-hand manipulation. For large android devices (e.g. tablets) this is a good approach but we consider it important that smartphones are adapted for comfortable one-handed use.

The Android design guidelines provided by Google (2012) mandates that designers place an action bar at the top of the screen. The action bar is supposed to contain important and commonly used functionality, but Park and Han (2010) found that the users in their study considered it inconvenient to reach for the corners of the screen, regardless of button size. While larger buttons considerably reduced the error rate for such reaching motions, user satisfaction remained low. Our dedication to both thumb-friendly interfaces and the guidelines forced us to make a choice about the action bar.

The greatest advantage of not following the design guidelines is that it frees us to design a UI that fits perfectly with the intended devices and users. With another design, buttons could be placed in a more ergonomic manner for thumb navigation. Another benefit is that the interface can be designed in a more recognizable way, differentiating 3Book from other competitors on the market.

In the end, it was decided to follow the design guidelines. Apart from being a recommendation from the developers of Android, there are several reasons for adhering to them. It makes the application feel “native” to the OS and users can easily understand the interface due to having seen the most important controls

in other apps. It also reduces time of development since accepting the guidelines is much faster then developing an alternative design. UI-elements such as the action bar are already implemented in the OS or available through third parties for legacy versions of Android, which further reduces development time. Finally, The data of Park and Han (2010) suggests that while users don't like reaching for the corners, they can still do so with reasonable speed and accuracy as long as the buttons are sufficiently large.

## 5.2 The state of the prototype

In the beginning of the project we tried to make a reasonable estimate of how many features could be completed during the course of the project. This estimate was largely guesswork since no project member had much experience with the technologies involved. As the project progressed a better picture of what each feature entailed in terms of programming was gained and the original estimate was found much too generous. A prioritization was done of the features, some of which were later implemented, some left as well-developed concepts and some as rough sketches. We nevertheless feel that, while not a complete application, the prototype to some extent displays all of the design principles discussed in this report.

Currently, the prototype can be used for finding and importing a book in the file system as well as reading a book with text and images straight from cover to cover. Only basic features of the book collection is implemented and meta-data presented about each book is limited to title and author. Continuing reading where one left of is not yet implemented. Neither is bookmarks or annotations. Simple preferences such as font or margin size, background/text color etc, can be changed but the reader doesn't change accordingly. The chapter-scroller has been usable but needs to be updated to work with the new rendering engine. Work has begun on many of these features and most are fairly close to completion, but at the time of writing there is still need of more work.

Issues like these were expected from the beginning of the project. The limited time meant a focus was put on creating a prototype that exemplifies our ideas and designs rather than a fully usable application. So while the application is far from complete, the goals of this project has been fulfilled.

# 6. Conclusions and Future Work

The purpose of this project was "to explore smartphone interaction design ideas for e-reading, and to develop a prototype that exemplifies these ideas". While not a complete application, we do think that 3Book has achieved this goal. It is, however, difficult to draw any solid conclusions about interaction design due to the exploratory nature of the project. User studies are needed to provide objective usability measures of the proposed design ideas.

With this caution in mind, we feel that our experience from this project allows us to make two tentative recommendations to smartphone application designers.

- Designers should take thumb-friendliness into account when they design their applications. We believe that adding redundant thumb controls is possible without bothering those who prefer to use two hands. This makes the application more accessible for those everyone who is either unable or unwilling to use two hands for interaction.
- Applications should be designed in a forgiving manner, recognizing that users are not always able or willing to devote their full attention or capabilities to the application. Interactions should provide a certain amount of slack so that high precision is not be required. Mistakes should be simple to avoid and easy to recover from.

## 6.1 Future research opportunities

The recommendations given above are given tentatively as they are not backed by any user studies. We think that the impact of thumb-friendly redundant controls could be an interesting area of study. Do redundant controls "cost" too much in terms of increased clutter and visual load? Are users that prefer two-handed interaction bothered by the presence of redundant controls? Are there applications that demand such precision that thumb controls cannot be used? These questions, and more, should be studied. It is encouraging that a number of papers address the issue of thumb-only interaction, but we hope to see more research done in this area.

Forgiving navigation is another interesting research subject that needs further exploration. Are there applications that become far less usable when the gesture space is simplified and restricted? Is a low error rate worth a potential increase in task completion time due to larger controls spread out over a wider area? Do power users feel less satisfied with applications that are less efficient, even if overall use is easier?

We would also like to see a user study performed with a navigational aid similar to our chapter scroller. How does such a hybrid menu perform in comparison to traditional structures of navigations?

Many of these questions will surely be answered as smartphones continue to grow in popularity and ubiquity. Smartphone interaction design is a fascinating topic that should face no problems in attracting researchers.

## 6.2 Future of the application

Getting from the current state of the prototype to an alpha release (with working preferences, meta-data, continue where left off and chapter-scroller) is estimated to require about a month of work from all three project members. At that point, the application would be a minimal but nicely designed e-book reader with the most unique feature being the chapter-scroller.

The application needs to expand its feature set to become a viable alternative that can compete with other e-readers in the market. The discrete views concept needs to be expanded with efficient views for tables, videos, programming code and aside content such as text boxes. Discrete views are especially important as it is a feature that sets 3Book apart from its competitors, unlike bookmarks or annotations that are more common (yet still planned for 3Book).

Other major future features include linux-like “workspaces” to facilitate viewing of multiple e-books simultaneously. Users should be able to switch between e-books using a gesture or a button. This should prove useful when cross-referencing between related books. Another planned feature if for users to be able to use links within the book to navigate between different sections. Navigational actions should be added to a history so that the device back button can facilitate a quick return to the original place in the book.

# Bibliography


Apple. Apple reinvents the phone with iphone. Press Release, jan 2007. URL `http://www.apple.com/pr/library/2007/01/09Apple-Reinvents-the-Phone-with-iPhone.html`.

Vinton Cerf, Yogen Dalal, and Carl Sunshine. *Specification of Internet Transmission Control Program*. The Internet Engineering Task Force, dec 1974.

Google. Android developers, January 2012. URL `http://developer.android.com/index.html`.

Andreas Holzinger. Finger instead of mouse: Touch screens as a means of enhancing universal access. In *Universal Access Theoretical Perspectives, Practice, and Experience*, volume 2615 of *Lecture Notes in Computer Science*, pages 387–397. Springer Berlin / Heidelberg, 2003. doi: 10.1007/3-540-36572-9_30.

A.K. Karlson, B.B. Bederson, and J.L. Contreras-Vidal. Understanding single-handed mobile device interaction. Technical report, Human-Computer Interaction Lab, University of Maryland, January 2006. URL `http://www.cs.umd.edu/localphp/hcil/tech-reports-search.php?number=2006-02`.

L. Kim and M.J. Albers. Presenting information on the small-screen interface: effects of table formatting. *Professional Communication, IEEE Transactions on*, 46(2):94 – 104, june 2003. doi: 10.1109/TPC.2003.813165.

Jennifer Lawinski. Two decades of e-reader evolution. Webpage, sep 2010. URL `http://money.cnn.com/galleries/2010/technology/1010/gallery.ereader_history/index.html`.

Marie Lebert. *Technology and Books for all*. University of Toronto, 2008. URL `http://www.etudes-francaises.net/dossiers/booksforall.pdf`.

Margus Pala. Phonegap performance measurement results, January 2012. URL `http://marguspala.com/phonegap-performance-measurement-results/`.

Pekka Parhi, Amy K. Karlson, and Benjamin B. Bederson. Target size study for one-handed thumb use on small touchscreen devices. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and*

*services*, MobileHCI '06, pages 203–210. ACM, 2006. ISBN 1-59593-390-5. doi: 10.1145/1152215.1152260.

Yong S. Park and Sung H. Han. Touch key design for one-handed thumb interaction with a mobile phone: Effects of touch key size and touch key location. *International Journal of Industrial Ergonomics*, 40(1):68 – 76, 2010. ISSN 0169-8141. doi: 10.1016/j.ergon.2009.08.002. URL `http://www.sciencedirect.com/science/article/pii/S0169814109001036`.

Phone-size.com. Phone size - phone size comparison made easy!, 2012. URL `http://phone-size.com`.

Annie Piolat, Jean-yves Roussey, and Olivier Thunin. Effects of screen presentation on text reading and revising. *International Journal of Human-Computer Studies*, 47(4):565 – 589, 1997. ISSN 1071-5819. doi: 10.1006/ijhc.1997.0145. URL `http://www.sciencedirect.com/science/article/pii/S1071581997901452`.

Brad Reed. A brief history of smartphones. *PC World*, 2010. URL `http://www.pcworld.com/article/199243/a_brief_history_of_smartphones.html`.

Andy Rubin. Android at mobile world congress: It's all about the ecosystem., feb 2012. URL `http://googlemobile.blogspot.se/2012/02/androidmobile-world-congress-its-all.html`. (Accessed 2012-05-03).

Christopher A. Sanchez and James Z. Goolsbee. Character size and reading to remember from small displays. *Computers and Education*, 55(3):1056 – 1062, 2010. ISSN 0360-1315. doi: 10.1016/j.compedu.2010.05.001. URL `http://www.sciencedirect.com/science/article/pii/S0360131510001235`.

Telecommunications. Product of the month: Bellsouth cellular/ibm release simon pda. *Telecommunications*, 28(1):116, January 1994. URL `http://research.microsoft.com/en-us/um/people/bibuxton/buxtoncollection/detail.aspx?id=40`.

C. P. Thacker, M. McCreight E, B. W. Lampson, R. F. Sproull, and D. R. Boggs. *Alto: A personal computer*. Palo Alto Research Center, 1979. URL `http://www.computer-refuge.org/bitsavers/pdf/xerox/parc/techReports/CSL-79-11_Alto_A_Personal_Computer.pdf`.

Jenifer Tidwell. *Designing Interfaces*. O'Reilly, 2005.

Wikipedia. Comparison of android devices, May 2012. URL `http://en.wikipedia.org/wiki/Comparison_of_Android_devices`.

# A. Statement of user requirements and acceptance criteria

## A.1 Introduction

This is a requirements specification document which will cover the most significant use-cases and functional requirements of the application. This document covers the goals of the final product. As the application is fairly advanced, there are no guarantees that all requirements will be fulfilled or that all use-cases will be achievable, due to time constraints.

### A.1.1 Purpose

The purpose of this document is to specify all functional-, non-functional- and quality requirements for the Android application 3Book. This document also covers project limitations and use-cases.

### A.1.2 Definition of customer and user group

This application is intended for everyone in possession of an Android smartphone that runs an operating system of version 2.2 or later.

## A.2 Use cases

### A.2.1 Add to collection

| | |
|---|---|
| **Goal:** | Add a book to the application's book collection(database). |
| **Main actor:** | User. |

**Main flow**

1. *Assumption:* User views the start screen.
2. User touches the "Open/Import" menu option.
3. Open import activity is displayed, showing a file browser.
4. User navigates to a folder containing e-books.
5. User touches one book.
6. Application displays a popup asking if user would like to open the book or import the book. The popup allows the user to set a default action.
7. User selects the import option.
8. Application displays a progress-bar dialogue while adding the book to the application database.
9. Application displays a toast notification when the operation is complete.

**Alternative flow**

**5-7:**

1. User selects multiple books.
2. Application displays multi-select related icons in the bottom of the GUI.
3. User touches import option.

### A.2.2 Open book

**Goal:** Open a book and begin to read.
**Main actor:** User.

**Main flow**

1. *Assumption:* User views the start screen.
2. User touches the "Open/Import" menu option.
3. Open import activity is displayed, showing a file browser.
4. User navigates to a folder containing e-books.
5. User touches one book.
6. Application displays a popup asking if user would like to open the book or import the book with a choice to set a default action.

7. User selects the import option.
8. Read activity is displayed, displaying a progress-bar dialogue while the book is loading.
9. Application displays the first page of the book.

**Alternative flow**

**2-7:**

1. User touches "My Collection" menu option.
2. Collection activity is displayed, showing a list of books.
3. User touches a book in the list.

### A.2.3 Change page using gestures

| | |
|---|---|
| **Goal:** | Be able to flip page backwards or forward with different types of gestures. |
| **Main actor:** | User. |

**Main flow**

1. *Assumption:* User has opened a book.
2. *Choice:* User performs a navigation action.
   (a) User uses a finger to swipe the page in left or right direction.
   (b) User touches within 0-33% or 66-100% of the screen width.
   (c) User uses volume buttons to change page.
3. Application shows the next page with an animation or by actively attaching a new page to user's finger while swiping.

### A.2.4 Open read overview

| | |
|---|---|
| **Goal:** | Display menu options, chapter menu and basic settings while reading. |
| **Main actor:** | User. |

**Main flow**

1. *Assumption:* User has opened a book.
2. User touches the menu hard-key.
3. Application shows the action bar, status bar and a chapter selection list. Chapters in the chapter scroller are represented as an scaled down versions of the first page in each corresponding chapter. The book text becomes covered by a black shape with some transparency.

**Alternative flow**

**2:**

1. User touches within 33-66% of the screen width.

### A.2.5 Change chapter

| | |
|---|---|
| **Goal:** | Change the current book chapter to another one. |
| **Main actor:** | User. |

**Main flow**

1. *Assumption:* User has opened a book.
2. User opens the reading overlay (see previous use case).
3. Application shows the overlay.
4. User scrolls the list and touches a chapter.
5. Application displays a progress dialogue while loading chapter.
6. Application displays the chapter when done loading.

**Alternative flow**

**4:**

1. User touches the "more" icon from action bar to display the menu.
2. Application displays read activity menu.
3. User touches "Table of contents" option.
4. Table of contents activity is displayed, showing a list containing all the chapters of the book.
5. User touches a chapter.
6. Read activity is displayed.

### A.2.6 Use dedicated views

| | |
|---|---|
| **Goal:** | View different types of content in corresponding dedicated activities. |
| **Main actor:** | User. |

**Main flow**

1. *Assumption:* User has opened a book and navigated to a page with a dedicated view element such as an image (the application will only support some types of view elements).
2. User touches view element.
3. Application launches the dedicated view activity.

### A.2.7 Add bookmark

| | |
|---|---|
| **Goal:** | Add a bookmark to the current page. |
| **Main actor:** | User. |

**Main flow**

1. *Assumption:* User has opened a book.
2. User opens overlay menu.
3. Application displays overlay menu.
4. User touches "bookmark" action icon from the action bar.
5. Overview closes and bookmark graphic slides down from the top of the current page. A bookmark is written to the application database.

### A.2.8 Adjust different settings

| | |
|---|---|
| **Goal:** | Change different application settings. |
| **Main actor:** | User. |

**Main flow**

1. *Assumption:* User views the start screen.
2. User touches the settings menu.
3. Application displays settings activity.
4. User makes adjustments.
5. Application saves adjustments.

**Alternative flow**

**2:**

1. *Assumption:* User displays an activity a with settings icon in action bar.
2. User touches settings icon from the action bar.

## A.3 Functional requirements

The functional requirements are specified as follows:

**Priority 1** Feature must be implemented for basic application functionality.

**Priority 2** Feature should be implemented to provide use value.

**Priority 3** Feature is desired to improve usability.

**Priority 4** Feature could be implemented as time permits to increase functionality.

### A.3.1 Browse for books

| | |
|---|---|
| **Requirement:** | User should be able to browse the file system to open a book. |
| **Priority:** | 1. |

### A.3.2 Open a supported e-book for reading

| | |
|---|---|
| **Requirement:** | Open a e-book that is supported by application. |
| **Priority:** | 1. |

### A.3.3 Page changing

| | |
|---|---|
| **Requirement:** | Change page by using gestures, touches or buttons. |
| **Priority:** | 1. |

### A.3.4 Change chapter

| | |
|---|---|
| **Requirement:** | Change chapter by using chapter selector list, table of contents view or by viewing last page in current chapter). |
| **Priority:** | 1. |

### A.3.5 Compatibility

**Requirement:** Application should be compatible with Android 2.2 and subsequent releases.
**Priority:** 1.

### A.3.6 Display images

**Requirement:** Show images and other graphical content in a book.
**Priority:** 2.

### A.3.7 View content in dedicated activity

**Requirement:** Use the dedicated views to explore different types of book content such as images and tables.
**Priority:** 2.

### A.3.8 Adjust settings

**Requirement:** User should be able to change some basic application- and reader settings such as font-size and margins.
**Priority:** 2.

### A.3.9 Add book/books to collection

**Requirement:** Use the file browser to add books to book collection.
**Priority:** 3.

### A.3.10 Multi language support

**Requirement:** Support for multiple languages.
**Priority:** 3.

### A.3.11 Widget

**Requirement:** Add a book widget to the desktop.
**Priority:** 4.

## A.4 Non-functional requirements

The non-functional requirements are specified as follows:

**Priority 1** The requirement must be satisfied for the application to be used commercially.

**Priority 2** The requirement is desired be fulfilled to increase the application usability.

### A.4.1 Consistency

| | |
|---|---|
| **Requirement:** | Elements that look the same, should act the same. Use same layout design, margins, text sizes etc throughout the entire application. |
| **Priority:** | 1. |

### A.4.2 Performance

| | |
|---|---|
| **Requirement:** | Application should run smoothly and heavier operations should be should be run in separate threads to prevent UI lock-ups. |
| **Priority:** | 2. |

### A.4.3 Maintainability

| | |
|---|---|
| **Requirement:** | All code should be well commented (javadoc is not required). The code will primarily follow Android conventions and secondarily Java conventions. |
| **Priority:** | 2. |

### A.4.4 Usability

| | |
|---|---|
| **Requirement:** | Interactions with the application should provide visual feedback. |
| **Priority:** | 2. |

### A.4.5 Error handling

| | |
|---|---|
| **Requirement:** | All exceptions should be properly handled. Errors should not crash application but show an error dialogue box instead. |
| **Priority:** | 2. |

## A.5 Quality standards

### A.5.1 Code design

- The majority of the code will practise Model-View-Controller architectural pattern.
- Package structure should be well organized.

### A.5.2 User interface

- Application should practice the Android 4.0 design guidelines where possible.
- A "Less is more"-approach should be followed throughout the entire application.