A5: Mobile Widgets (Java/Android) -- PDF Reader

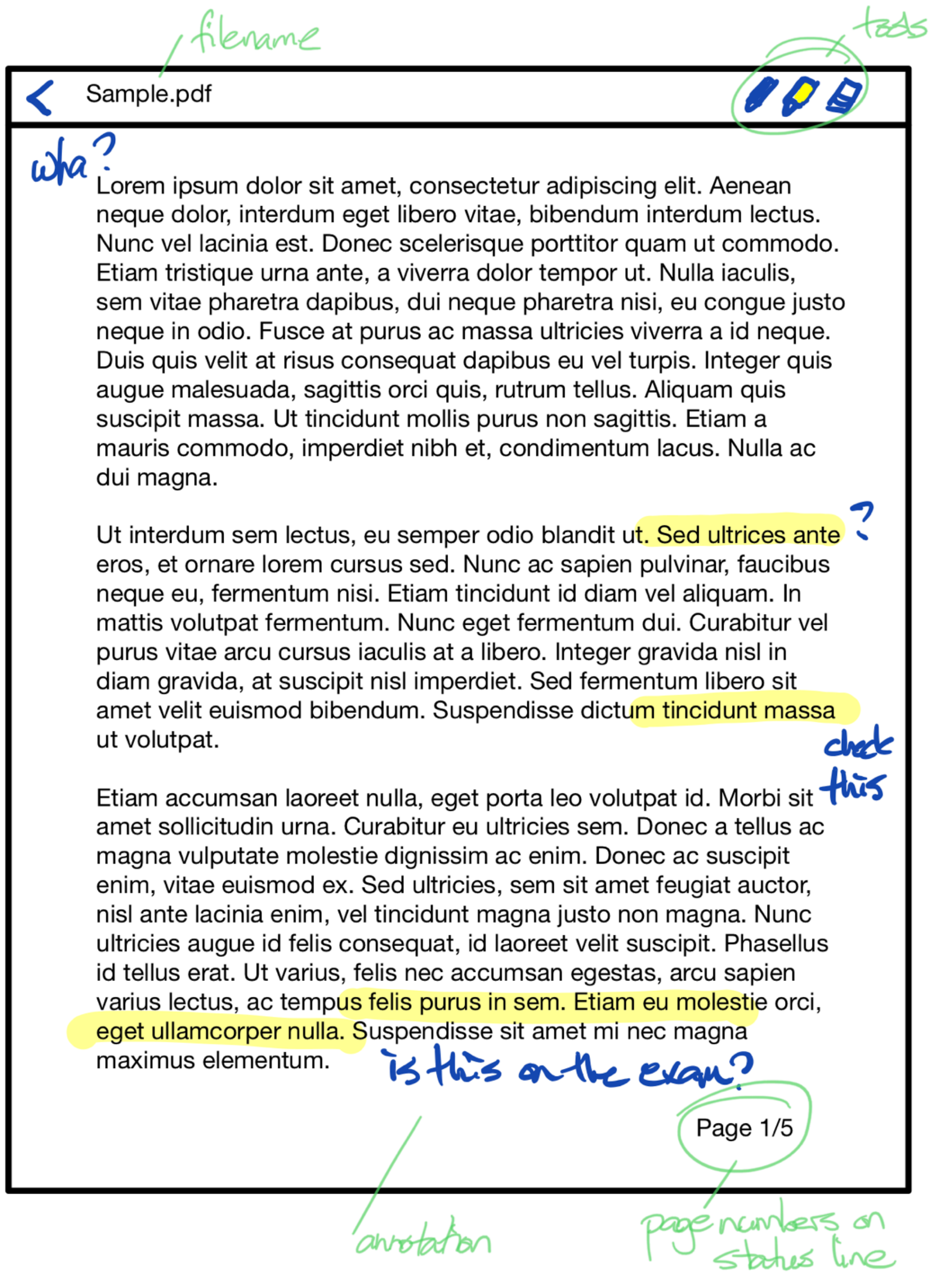
Synopsis

You will design and implement a PDF reader that allows a user to read and annotate a document on an Android tablet.

Requirements

General

Design and implement a mobile PDF reader on Android. The screen should look something like this. Note that you have license to change the layout and appearance, as long as you support all of the features and functionality described below.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean neque dolor, interdum eget libero vitae, bibendum interdum lectus. Nunc vel lacinia est. Donec scelerisque porttitor quam ut commodo. Etiam tristique urna ante, a viverra dolor tempor ut. Nulla iaculis, sem vitae pharetra dapibus, dui neque pharetra nisi, eu congue justo neque in odio. Fusce at purus ac massa ultricies viverra a id neque. Duis quis velit at risus consequat dapibus eu vel turpis. Integer quis augue malesuada, sagittis orci quis, rutrum tellus. Aliquam quis suscipit massa. Ut tincidunt mollis purus non sagittis. Etiam a mauris commodo, imperdiet nibh et, condimentum lacus. Nulla ac dui magna.

Ut interdum sem lectus, eu semper odio blandit ut. Sed ultrices ante eros, et ornare lorem cursus sed. Nunc ac sapien pulvinar, faucibus neque eu, fermentum nisi. Etiam tincidunt id diam vel aliquam. In mattis volutpat fermentum. Nunc eget fermentum dui. Curabitur vel purus vitae arcu cursus iaculis at a libero. Integer gravida nisl in diam gravida, at suscipit nisl imperdiet. Sed fermentum libero sit amet velit euismod bibendum. Suspendisse dictum tincidunt massa ut volutpat.

Etiam accumsan laoreet nulla, eget porta leo volutpat id. Morbi sit amet sollicitudin urna. Curabitur eu ultricies sem. Donec a tellus ac magna vulputate molestie dignissim ac enim. Donec ac suscipit enim, vitae euismod ex. Sed ultricies, sem sit amet feugiat auctor, nisl ante lacinia enim, vel tincidunt magna justo non magna. Nunc ultricies augue id felis consequat, id laoreet velit suscipit. Phasellus id tellus erat. Ut varius, felis nec accumsan egestas, arcu sapien varius lectus, ac tempus felis purus in sem. Etiam eu molestie orci, eget ullamcorper nulla. Suspendisse sit amet mi nec magna maximus elementum.

When first launched, the application will load a sample PDF that you have included in your project. The titlebar should show the name of the PDF. The statusbar should show the current page number and the total number of pages (e.g. "page 2/5").

You need to support the following functionality. Unless a feature specifically states how you should activate it, you can choose how to present these to the user. e.g. buttons on toolbar, floating buttons for page up/down, or 'two-finger swipe left' to undo.

- **Undo/Redo**: Undo the last change that was made to the document. The user should be able to undo at least the last 5 actions that were performed. Redo can be used to revert the undo (as described in class).
- **Drawing**: The user can write on the screen with a thin line in blue ink, allowing them to write notes or annotate the document. The user cannot change the color or thickness of the line, so select line characteristics that is appropriate for writing on the PDF.
- **Highlighting**: The user can draw over the existing document with a thick, transparent yellow brush that allows the user to highlight the text in the PDF. Make sure that the highlighter is transparent enough that the text beneath it remains visible! See the diagram above for an example.
- **Erase**: The user should be able to erase an existing drawing or highlighting (note that 'undo' does not replace this feature).
- **Zoom & Pan**: The user can use two fingers to zoom-in and zoom-out (by bringing their fingers closer together or spreading them apart over the area of interest). When zoomed-in, users can pan around to reposition the document. These gestures should behave the same as standard pan-and-zoom (hint: try Google Maps on the emulator to see how these gestures should work). Users can draw and highlight a document at any scale, and the annotations should scale with the document canvas.

You should support multi-page PDFs, and your sample document needs to have at least 3 pages. The user should be able to scroll between pages - you need to decide how to add functionality to support this (up/down buttons, or perhaps a custom gesture).

Any gestures that you implement (e.g. zoom, pan, possible moving between pages) should respect direct manipulation principles. For instance, content should remain under your finger as you zoom or pan. Gestures should also follow principles discussed in class (e.g. given design principles, swipe-back would be a good choice for navigating to the previous page, but a poor choice for the erase tool).

You should optimize for a tablet in portrait mode (see below for device/AVD details). You should disable orientation changes in your application so that the layout remains in portrait mode as the device is rotated.

You should save changes when the user changes pages, or exits. A user should be able to make changes, exit your applicaton, and then reload it and see their changes intact. Drawing and highlighting should be saved on each page i.e. it should not be lost when the user navigates between pages.

Technical Requirements

- You are provided starter code that loads a sample PDF and logs touch events (a5starter.zip). You are free to use this code, or ignore it and implement everything your own way (but using this code will likely save you some time!)
- You must create this assignment as an Android project using IntelliJ 2019.3, with the Android 10.0 (API 29) SDK. Instructions to install and configure IntelliJ are included in the Android development slides.
- For development and testing, use a Pixel C tablet with API 29, in vertical orientation.
- You can use any functionality included in the Android SDK, and are free to use code snippets from the Android sample code included with the SDK (i.e. a snippet being a small segment of code). You can also use any code that we have placed on the public CS 349 Git repo. If you do this, put comments in your code referencing the original source. You are not allowed to use any other third-party code or libraries, unless you obtain permission from the instructor (permission on Piazza is fine).
- You are not required to use Model-View-Controller, but you *must* save your data in such a way that no data is lost.

Submission

Your directory structure for your assignment submission should be in subdirectories under your a5/ directory in your Git repository. It should use a standard file layout for an Android IntelliJ project.

Your submission needs to include:

- All source code and resources required to build and execute your program.
- An IntelliJ project that compiles everything using the specified JDK and Android SDK, and which has a Run target that will execute your program.
- A `readme.md' file with any details required to help the TA grade your assignment.
- An APK file at the top-level of your directory structure. You will likely need to manually create this (Build -> Build APK) and copy to this location. This will help the TA to load and test your application without needing to load the project.

Your `readme.md' file needs to include, at a minimum:

- Your name,
- Your student number,
- Your operating system & version,
- The version of Java and the Android SDK that you used,

- The source for your image assets, as required.

Assessment

Late assignments will not be accepted. Markers will test your application by running your APK file against the appropriate AVD. Your submission will be assessed roughly as follows:

5%
Code compiles and runs, readme, working project.
15%
PDF is displayed. User can scroll between pages.
15%
User can annotate the PDF using either the pen or highlighter.
15%
User can erase an annotation using the erase tool.
15%
User can undo/redo up to 5 levels of actions that they perform.
15%
User can zoom and pan using standing gestures. Annotations scale with the canvas.
10%
Data persists when application is restarted, or when the user nagivates between pages.
10%
User controls are obvious/clear and intuitive, and support the features required.

Versions

1.0. Mar 20, 2020. Published.
