
GitHub Username: AppScientist

Offline Reader

Description

There are many instances where we come across an interesting article online, and decide to read about it later. And soon we would forget about the article. With this app, articles can be saved for later viewing. Articles once saved can be viewed without internet connection.

Articles can be easily added by pressing the share button in the browser and selecting “Offline Reader”.

Problem solved is an extra feature for rich sharing of content. Using the latest Nearby messages API, we can directly share the articles to other devices.

Intended User

Intended for everyone especially with countries with slow internet connectivity.

Features

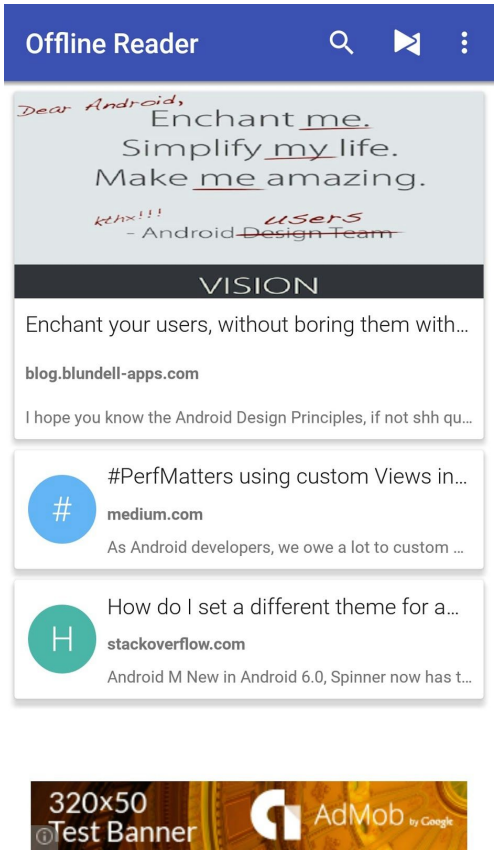
Main features of your app.

- Saves articles for offline reading
- Search for saved articles.
- Text to Speech feature to read articles.
- Share the saved articles directly from one device to another.

User Interface Mocks

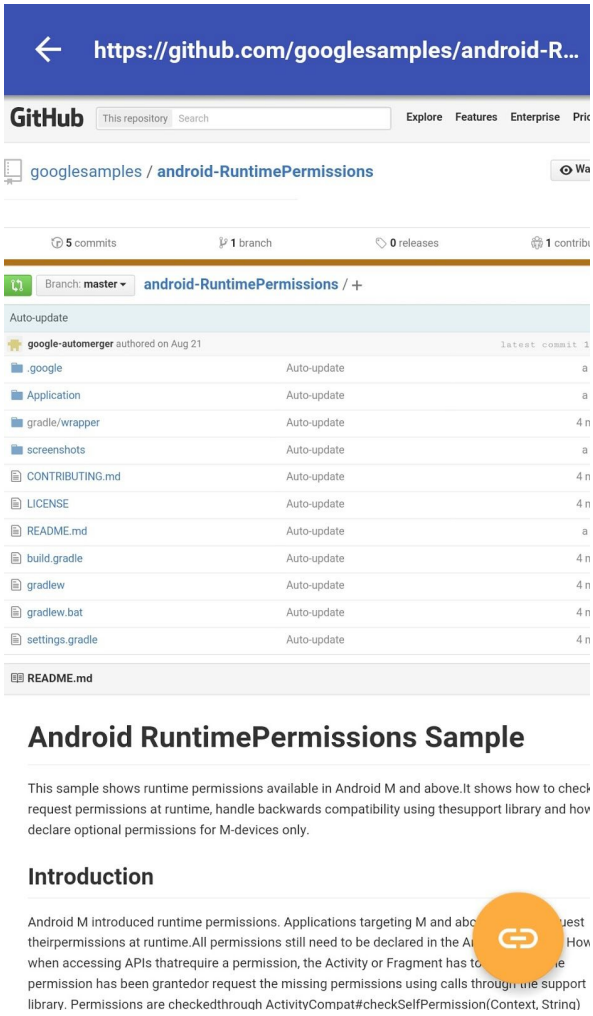
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Screen 1



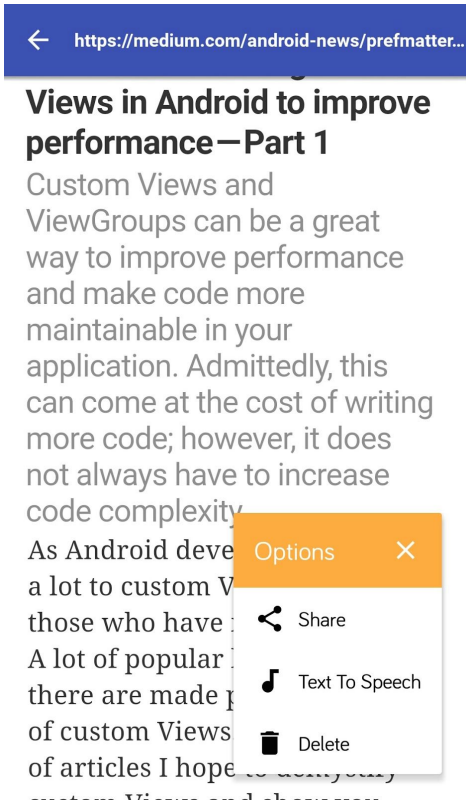
Main screen displaying saved article. Search icon to search for particular article in list. Nearby icon to start sharing of data.

Screen 2



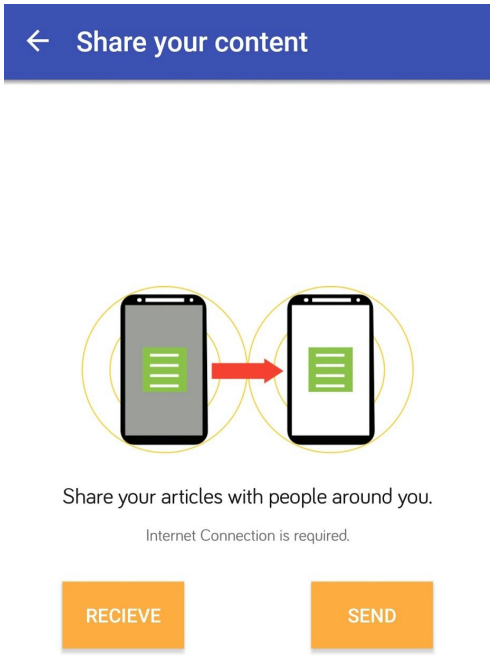
WebViewActivity displaying the contents of the a selected article.

Screen 3



FAB transformed to sheet and displaying options.

Screen 4



NearbyMessagesActivity showing the intro page.

Key Considerations

How will your app handle data persistence?

Articles name, url and description are stored in SQLite Databases and web content are saved in internal memory for viewing later.

Describe any corner cases in the UX.

NA

Describe any libraries you'll be using and share your reasoning for including them.

Glide to handle the loading and caching of images.

TextDrawable to get gmail-like list.
Fst for fast serialization of data.
Jsoup for easy parsing of HTML content.
AppIntro for making an easy guide on how to use the app.
Fabtransformation for an easy way to implement a FAB to sheet transformation animation.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

- Configure MinSdk as 16 and TargetSdk as 22
- Create activity with blank fragment.
- Configure all required third party libraries in gradle file.
- Enter the Nearby messages and Readability API key in gradle.properties and define them in gradle file.
- Include required Roboto font files in asset folder.
- Set up the styles.xml file.Add a Theme.AppCompat.Light.NoActionBar and set all necessary colorPrimary, colorPrimaryDark and colorAccent colors.
- Add the action.Send intent filters in manifest, so that articles can be added to the app by pressing the share button in the browser.

Task 2: Implement UI for Each Activity and Fragment

- Firstly implement a splash screen, so that as the app starts running users will not see a blank page for few seconds. Using sketch create a simple logo which describes the app.
- Next implement an IntroActivity using AppIntro library which will be a guide for the users on how to use the app.
- Next implement MainActivity and fragment. This is simple UI with a toolbar and recyclerview. RecyclerView displays all the added articles.
- Next implement UI for activity and fragment for displaying article content.This UI contains toolbar ,webview and a FAB.
- Finally UI for Nearby messages. UI will contain a brief explanation and send and receive button. On clicking send button, UI will contain the list of shared articles which will be sent. On clicking receive button, UI will be blank screen. It will contain data once items are received.

Task 3: Implement Storage

- Implement a SqilteDatabase and content provider for storing article related data.
- Declare the content provider in manifest.

Task 4: Implement Intent Service

- Implement intent service to download the article title, url description and image.
- Store these data in SQLite through content resolver.

Task 5:Implement Loaders

- Define a recyclerView and implement its adapter.
- Implement loaders in fragment and through loaders send the cursor to the recyclerview adapter to display the added articles.

Task 6: Implement Swiping and Multi Touch for RecyclerView

- Implement ItemTouchCallback for swipe to delete action.
- Implement multitouch for selecting multiple items at once.

Task 7: Implement SearchView

- Implement searchview to search for articles in saved list.

Task 8: Implement WebViewActivity

- Implement a custom WebViewClient to load the saved articles and images.
- Implement various FAB features- save articles to list, delete article, text to speech and share.

Task 9: Implement HeadlessFragment

- Implement a fragment which contains no UI. This fragment will be used as a connection helper.It will connect to GoogleApiClient.

Task 10: Implement NearbyMessagesActivity

- Once connected to GoogleApiClient, implement the nearby messages features to share and recieve articles between devices.