GitHub Username: bl1b

Company Feed Library

Description

The “Company Feed Library” is an easy to use Android-Component that companies can use to bundle their various sources of information such as RSS-Feeds, YouTube-Channels, Facebook-Pages and Tweets. The library is highly configurable so that new sources can be added with ease. For each sources’ entry there’s an overview item displaying the basic information such as an image, title and description. Each item has a detailview depending on the the source’s type. Interaction (such as commenting) on a social media source will be handled by their respective apps and/or webviews (depending on whether the specific app is installed or not). Direct interaction is on the roadmap for a future release.

Intended User

Intended user of the library is the user base of the company's existing application. Alternatively the library can be the sole content of a new application for a company.

Features

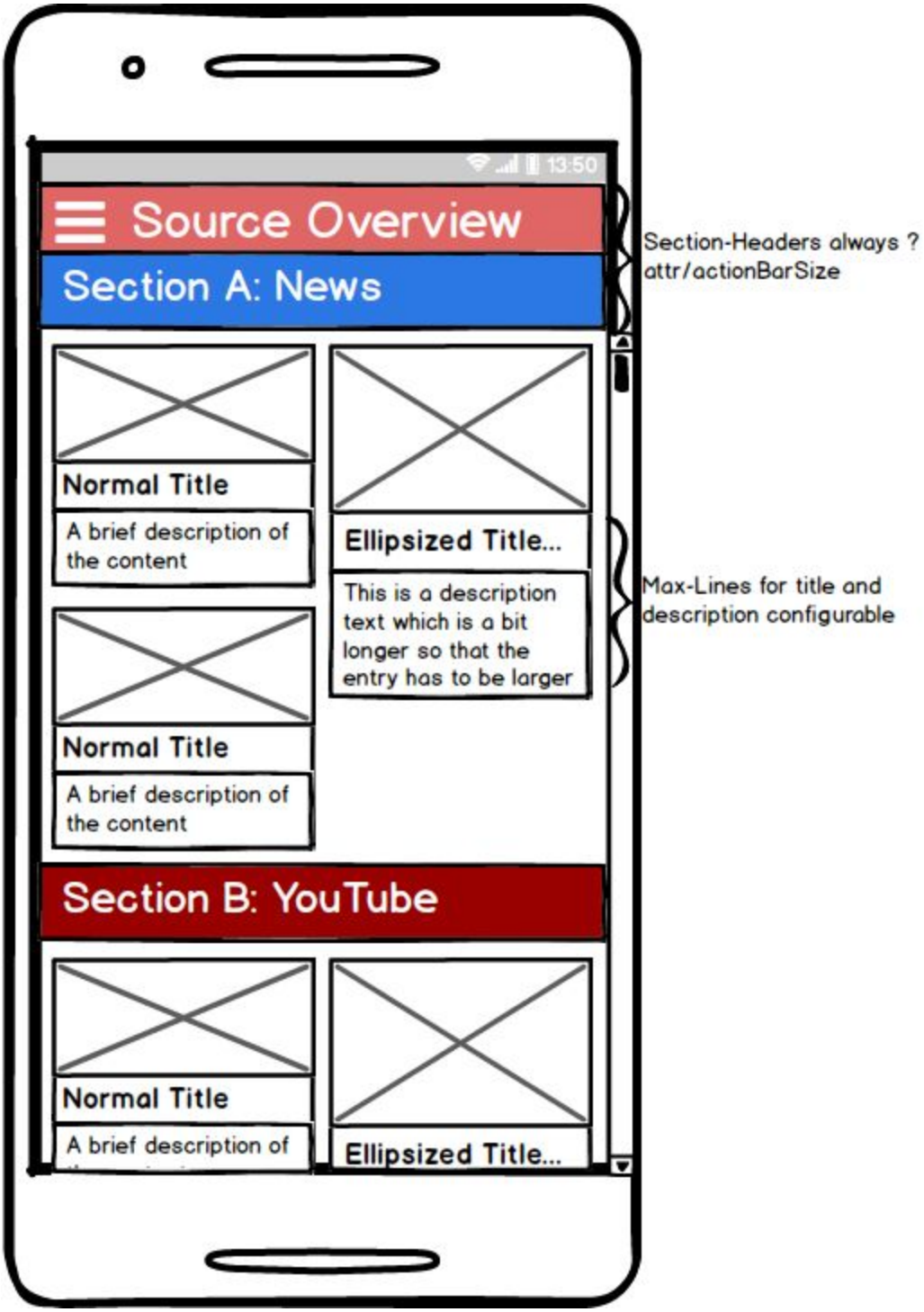
Features of the application:

- Displays an overview of a company’s information sources:
    - RSS-Feeds as JSON-Requests
    - YouTube-Content using the YouTube-API
    - Facebook-Content using the Facebook-API
    - Tweets using the Twitter-API
- Shows detailed information about the selected information source:
    - Longer Text with larger image and more information for RSS-Feeds
    - A YouTube-Video for YouTube
    - An enlarged view for Facebook
    - An enlarged view for Twitter
- Overview as Staggered-Grid-Layout on Smartphones (Portray only)
    - Expandable-Listview which can be used to show groupings of information inside a Navigation-Drawer
- Master-Detail on Tablets
    - Master: Expandable-Listview which can be used to show groupings of information sources
    - Detail: The overview for the selected grouping

- Notifications on updates
- Tracking of user interactions with Google-Analytics
- Optional integration of AdMob allowing the company to optionally show ads in their application

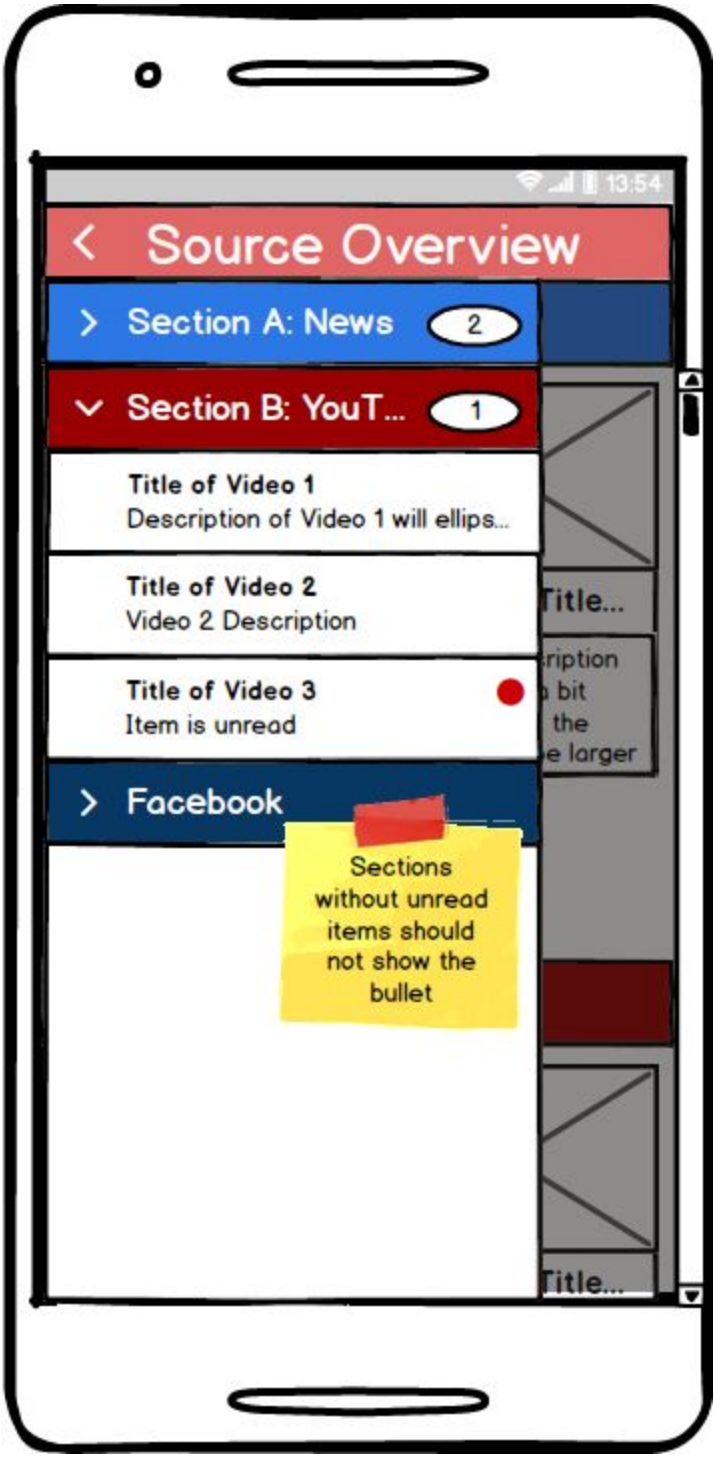
User Interface Mocks

Screen 1



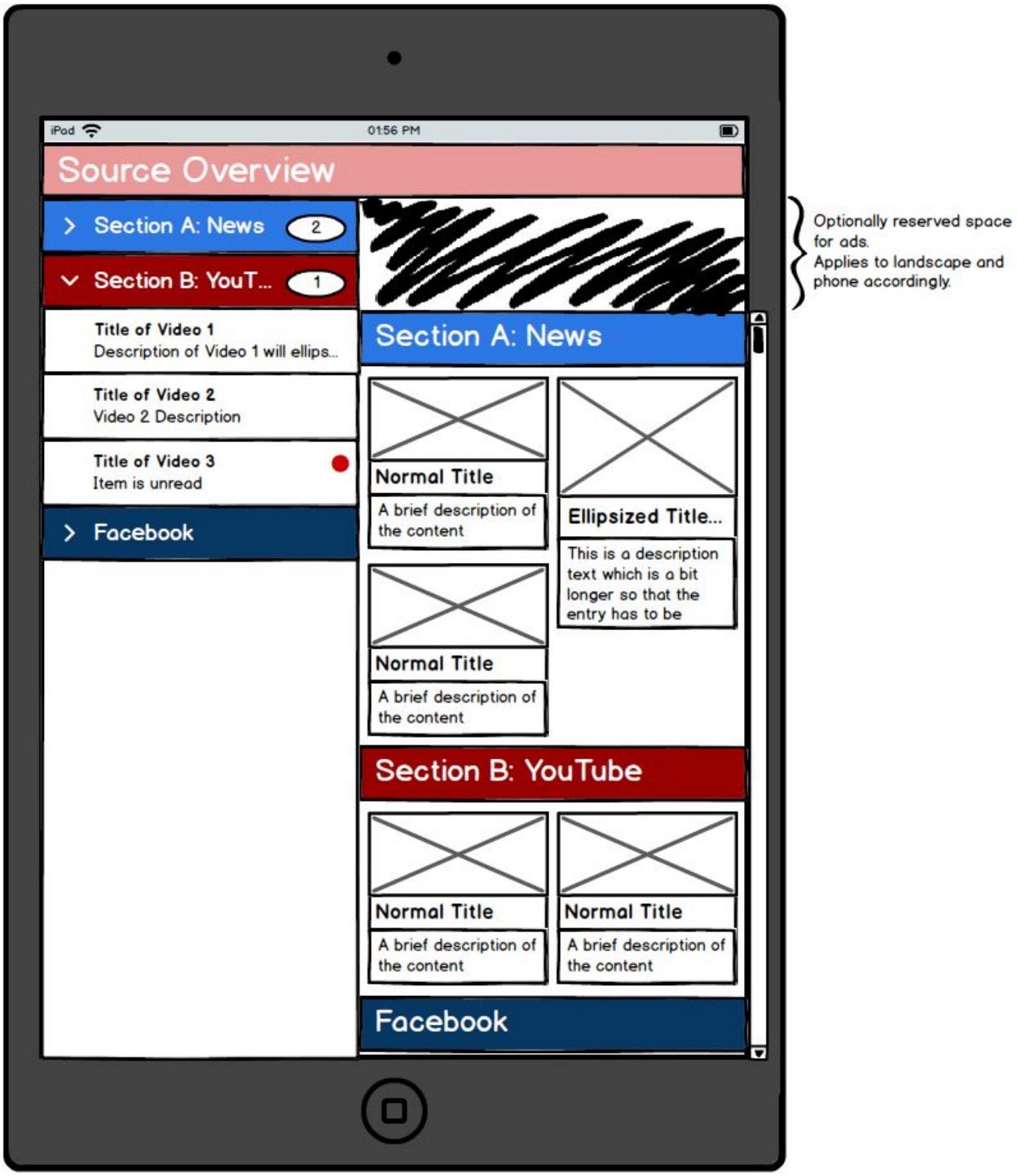
ListView for the item-overviews is a RecyclerView with a StaggeredGridLayout. The DrawerMenu is used to display an overview of available items and is replaced by the Master-View on Tablets.

Screen 2



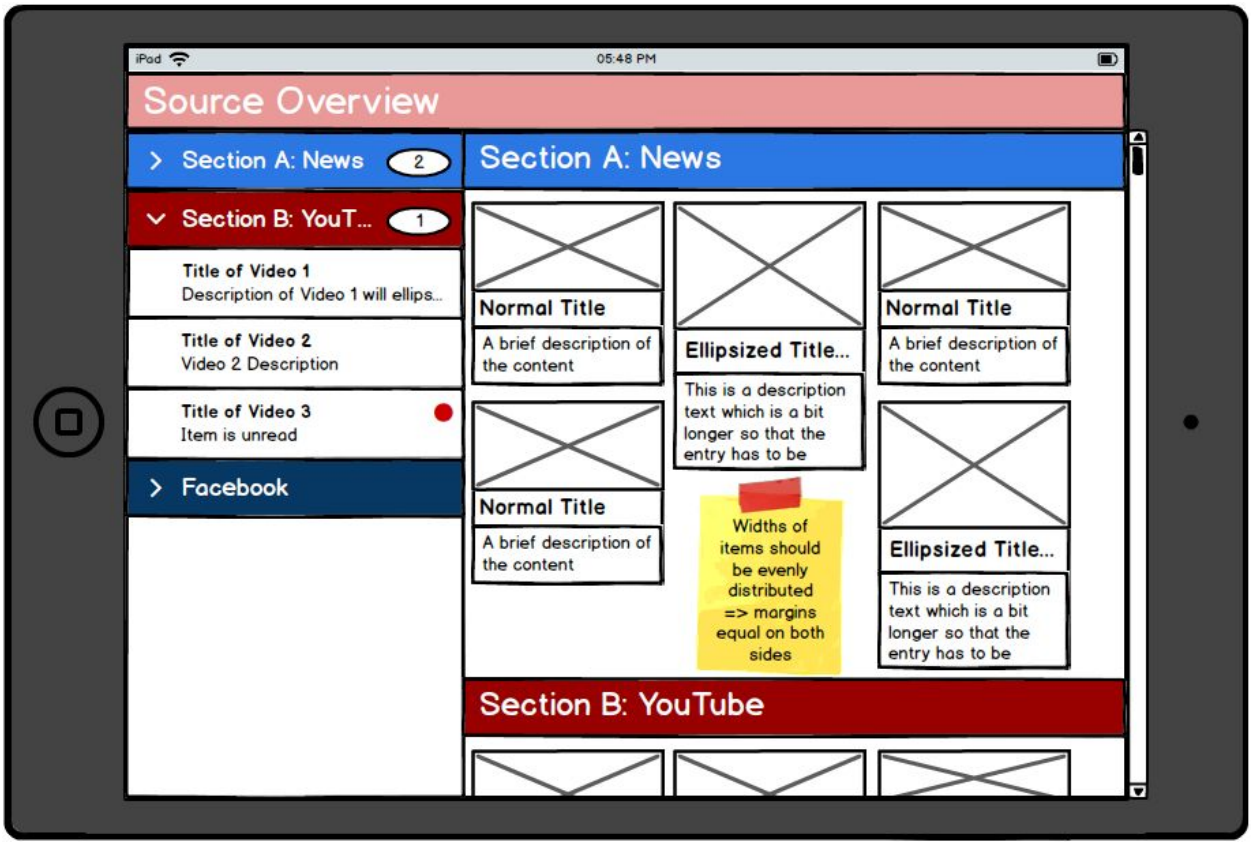
This is the section overview as DrawerMenu. If there are unread articles in a section they are marked as unread and a bullet on the section-header displays it.

Screen 3



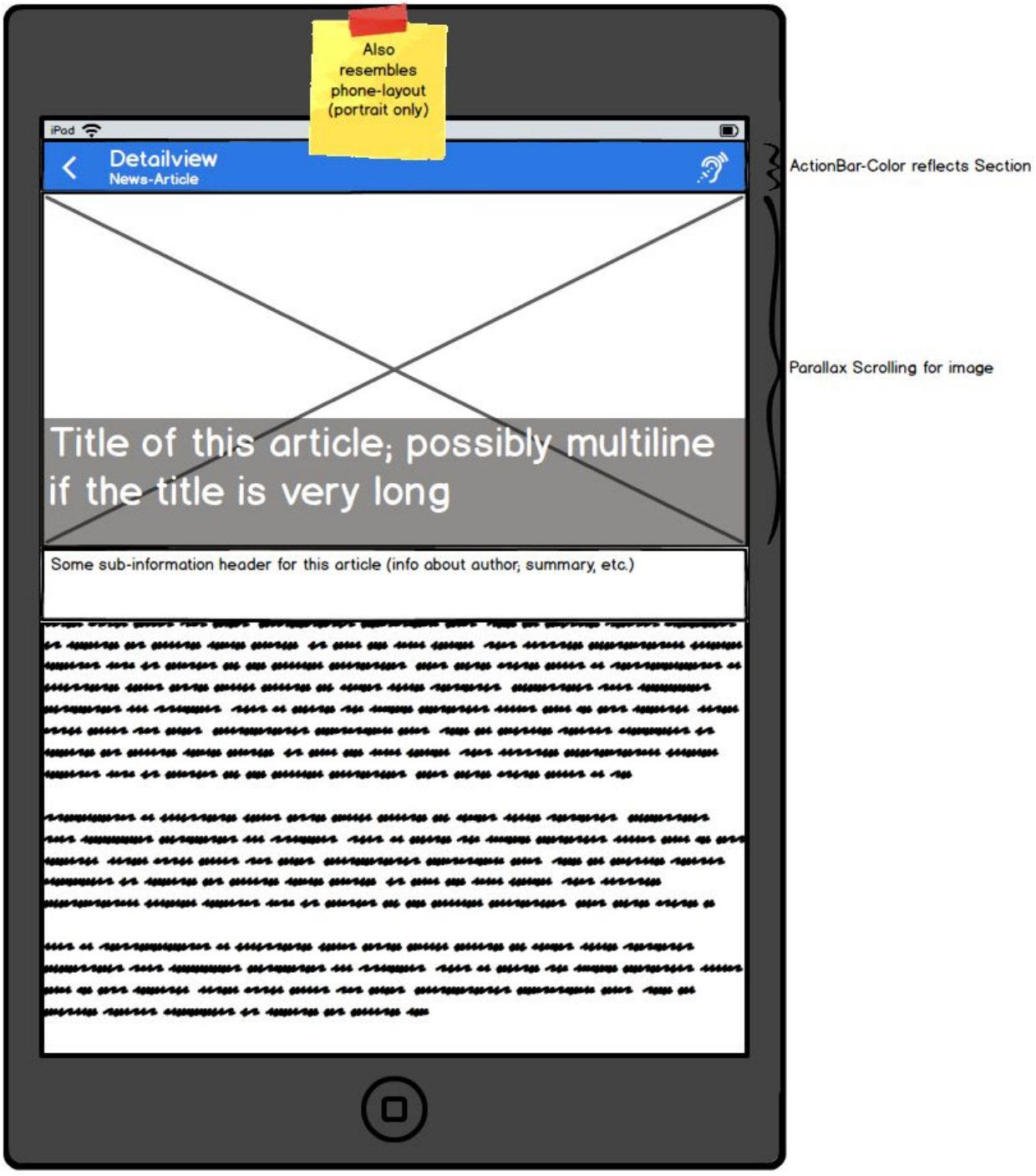
Master/Detail - View. Ad-Area above ListView not scrollable.

Screen 4



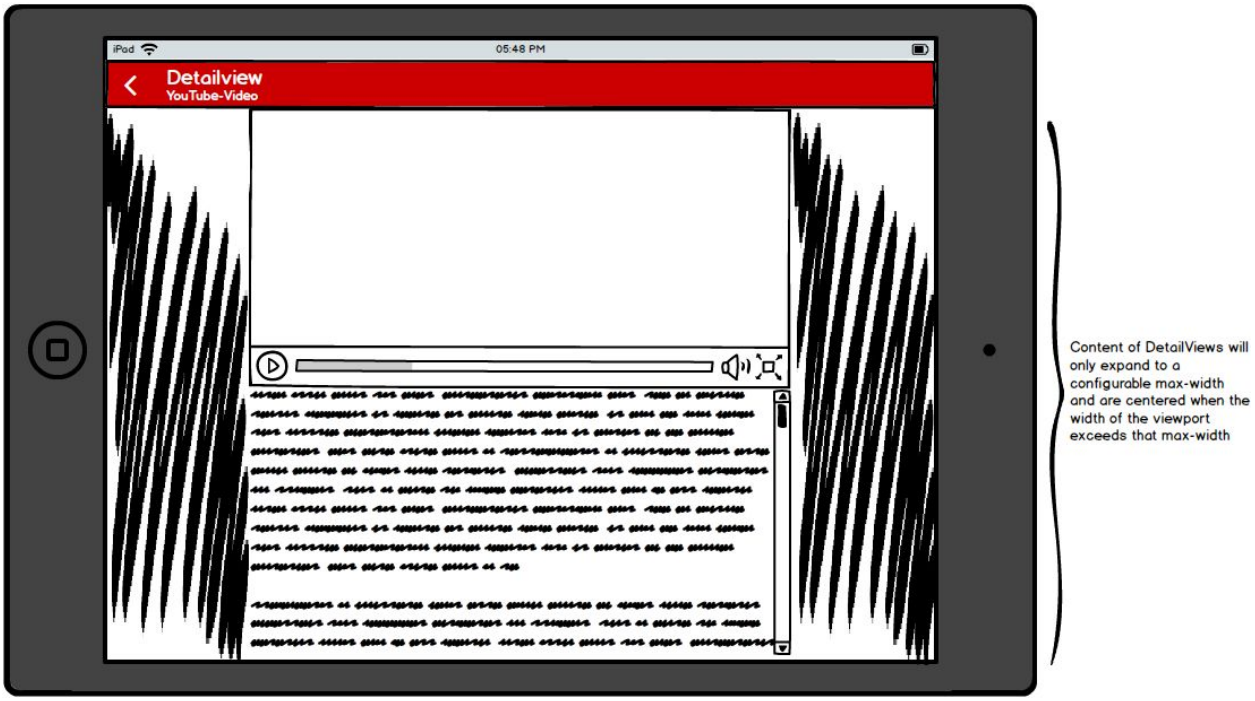
Pretty much the same as the portrait-layout but with more columns on the overview. The width of the column show be equally distributed considering the available width.

Screen 5



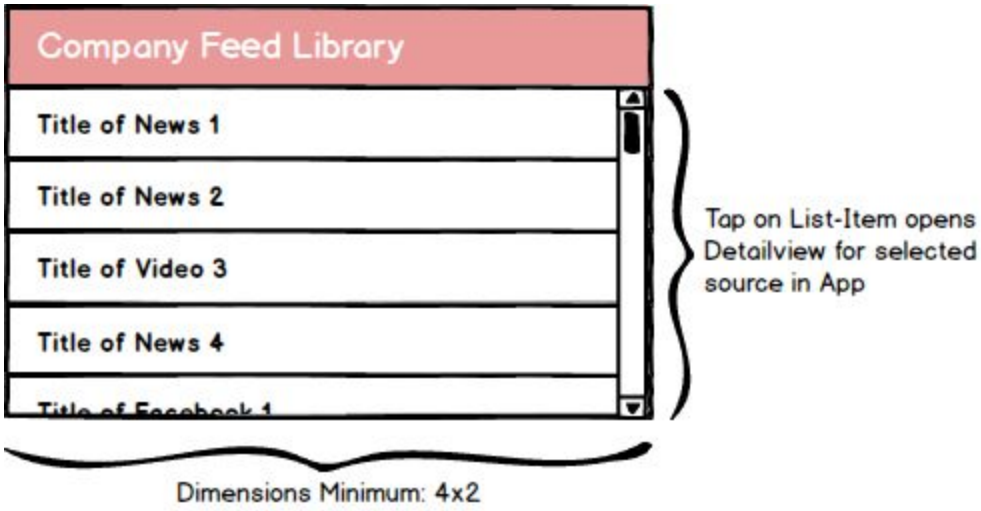
This layout also reflects the phone-layout (which is portrait only) and represents the detailed contentview of a RSS-Source (e.g. Newspaper-Article, Blog-Entry, etc.). The MenuItem represents the ability to you the system's TTS capability to read out the article to the user.

Screen 6



This shows the Detailview for a YouTube-Source. It also shows that on very large width the width of the content is limited and the content itself centered. This should prevent extreme changes to the aspect ratio.

Screen 7



This images shows the layout for the Application's Widget. It will have a minimum dimension of 4x2 showing the last `n` unread items. If the list is empty a message: 'There are no unread items' will be shown.

Key Considerations

How will your app handle data persistence?

Data will be stored locally in a realm-database (https://realm.io). Realm already provides Adapters for Listviews and/or RecyclerViews. A classic ContentProvider is not required; however a ContentProvider is required to power the Widget with data. The configured API's will be called using an IntentService which is started whenever the MainActivity is resumed and additionally will called on an interval using the AlarmManager. The database will be update inside the service and the updated state is passed to the UI using RXJava in order to update the Adapters (and therefore the UI).

Describe any corner cases in the UX.

In order to use the YouTube-API the YouTube-App needs to be installed. If the app is not installed Detail-Content of YouTube will be opened in an external web browser. This will be similar for Facebook and Twitter.

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

Picasso will be used for image loading combined with a Disk-based LRU-Cache so that images for items are stored on the device and available offline.
RXJava will be used for event-based data-handling (such as result-handling of services, database-updates, etc.).
Dagger2 will be used as dependency injection framework. Since the main intent of this project is to create a library modularity and extensibility are key. The application that this library may be included will also be using dagger.
Retrofit will be used for the JSON and XML-Requests on the sources' APIs. This will make the parsing and caching of data easier since this will be handled by the okhttp-backend.
OKHttp will be used for smaller http-requests (if necessary).

Describe how you will implement Google Play Services.

Analytics will be used to track the users' interaction within the application to identify hotspots which can should be targeted on further project iterations. It will also be used to track exceptions and crashes in order to detect issues for future bugfix releases.
AdMob will be used to give customers the ability to integrate ads in their otherwise free application.

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

Setup GitHub

- Add LICENSE file to existing GitHub-Repo (Repo has already been setup with this file)
- Add README.md file with a basic project description (including a link to this document)

Setup Project

- Clone GitHub-Repo into working folder
- Create new Android-Project with Blank Activity

Setup IDE

- Create appropriate copyright templates and file headers

Setup Gradle

- Include planned libraries into build.gradle
- Collect versions for libraries in a single gradle-file
- Setup proguard-file with the requirements of each library

Task 2: Setup test-data

In order to build the UI and for potential automated testing test-data are required.

- Collect some rss.json examples for articles
- Create offline response files for a YouTube API-Request
- Create offline response files for a Facebook API-Request
- Create offline response files for a Twitter API-Request

Task 3: Implement UI for Each Activity and Fragment

MainActivity UI

- Fragment for Overview (including a recyclerview with staggered grid layout)
- Fragment for Master-View (including ExpandableListView)
- DrawerMenu for Smartphone
- SettingsFragment for configuration measures

DetailActivity UI

- UI for RSS-Sources (Phone/Tablet Portrait, Tablet Landscape)
- UI for YouTube-Sources (Phone/Tablet Portrait, Tablet Landscape)

- UI for Facebook-Sources (Phone/Tablet Portrait, Tablet Landscape)
- UI for Twitter-Sources (Phone/Tablet Portrait, Tablet Landscape)

Task 4: Loading data from external sources

Implementation of API-Requests for each supported Source

- JSON-Requests for RSS
- JSON-Requests for Facebook (maybe XML depends)
- JSON-Requests for YouTube
- ???-Requests for Twitter

Task 5: Setup persistence layer

Store data in realm-database and issue events in order to update the UI.

Task 6: Use real data

Use the real data stored inside the database to populate the master-view (or drawermenu) as well as the recyclerview.

Task 7: DetailViews

Setup the detailviews to use the provided data and show their respective content.

Task 8: Transitions

Implement the transitions between overview and detailview.

Task 9: Widget

Implement Widget with Listview as seen on Screen 7. Implement a ContentProvider for the Widget using the Realm-DB as underlying Data-Backend using a MatrixCursor as query-result.

Task 10: Test

Test-Iterations for everything that is not tested via Unit-Tests and/or Integration-Tests. Run several iterations of monkey-tests.

Task 11: Release

Setup release-key and prepare signingConfig in build.gradle. Create APK and Upload.