How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

GitHub Username: rgr-myrg

Pocket Buddha

Description

Welcome to your guide to daily inspiration of the Buddha's words.

Pocket Buddha empowers you to read, listen, and study verses from the Pali Canon in both English and Pali. Each verse is accompanied by a beautiful image to inspire both learning and contemplation.

Search by title or text using your keypad or voice and save your favorite verses as you navigate effortlessly through each one. Or simply scroll through captivating Buddhist images and select which verse to read.

Pause a bit to listen to the Pali audio and reflect on the English translation.

Use the Pocket Buddha Widget to learn a new Pali word daily.

Pocket Buddha makes it easy to read, study, listen, organize and share the Buddha's words with friends.

Intended User

Pocket Buddha is intended for all interested in learning, studying, reflecting, and becoming inspired by the Buddha's words in daily life.

Features

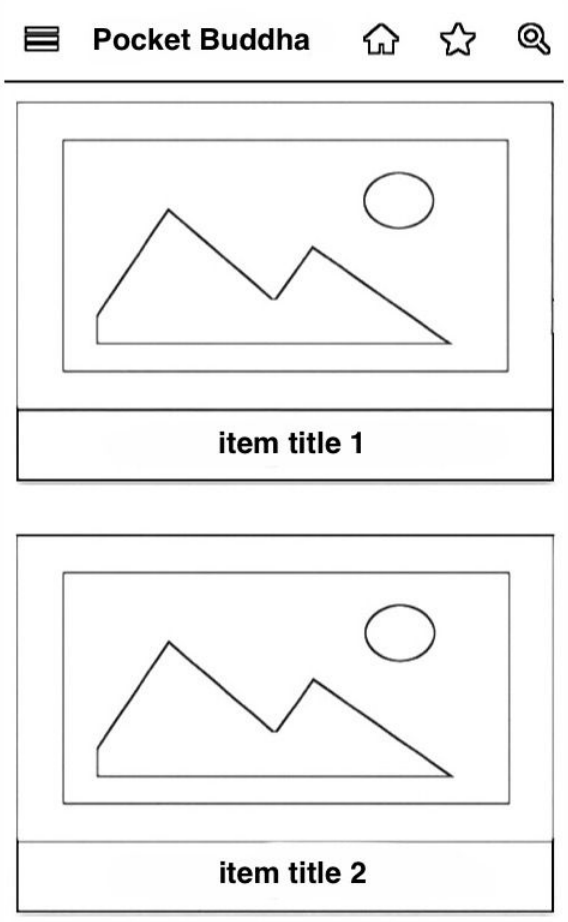
- Feed mash up of content with images.
- Endless scrolling when in list view.
- Swipe enabled navigation for content in detail view.
- Ability to save content to Favorites.
- Audio player for streaming mp3 content.
- Sharing functionality with chooser.
- Voice and text search enabled.
- Sorting of Favorites alphabetically or by date ascending and descending.
- Collapsing toolbar layout with parallax scrolling on images in detail view.
- Widget to display daily word.

- Floating action button menus.

User Interface Mocks

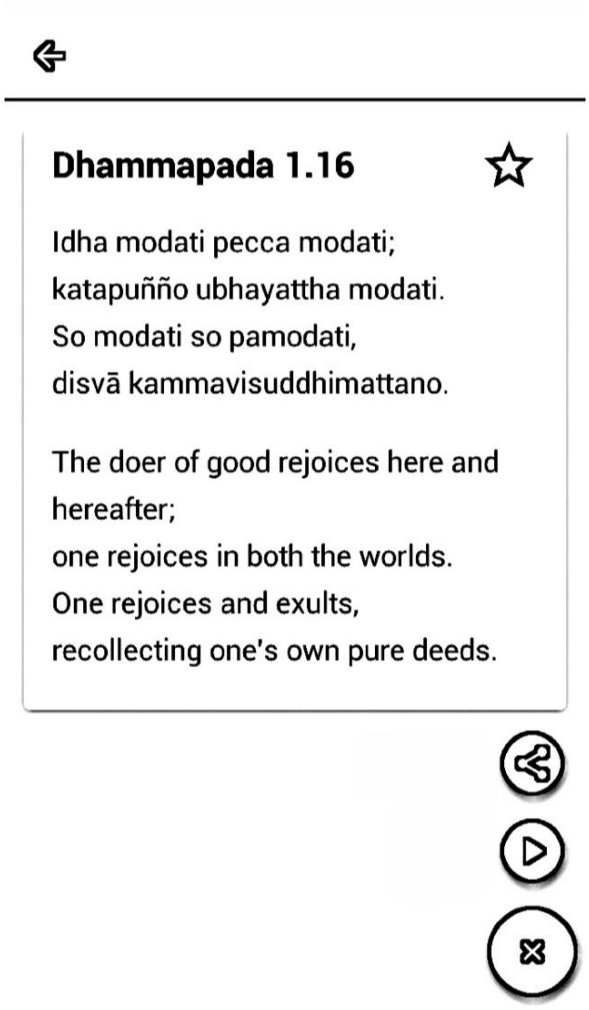
Screen 1

Home Activity screen displays a list of images and content title in a card view for available items in the feed. Users can scroll until feed items are exhausted. Clicking on items will start a Detail Activity.



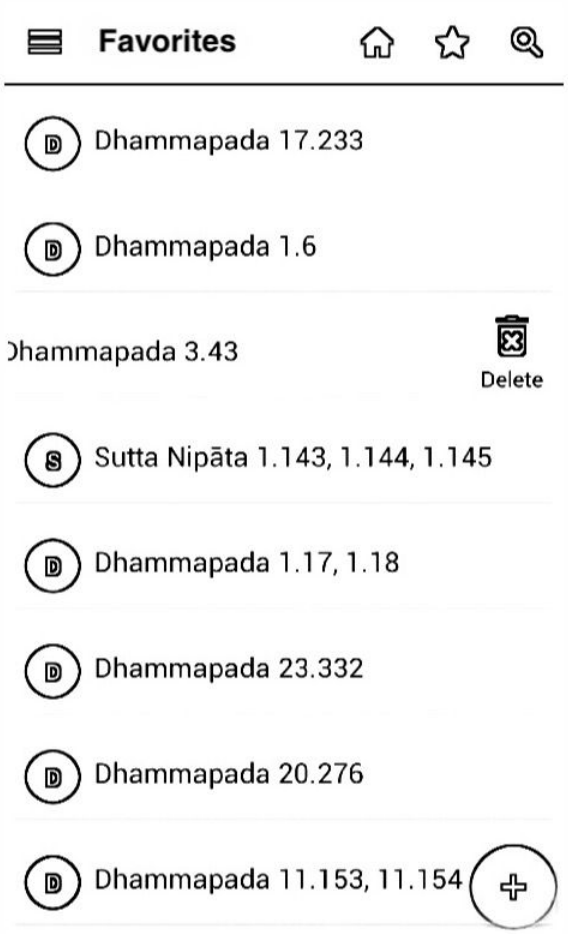
Screen 2

Detail Activity view of content. Users can swipe right or left to navigate through items. Clicking on the star icon saves the item to the database. Conversely, deselecting a starred item will delete the item from the database. A floating action button reveals additional buttons to share or play the sound associated with the content. Clicking on each button performs the corresponding action.



Screen 3

Favorites Activity screen where a list of saved items is displayed. Clicking on an item will initiate a Detail activity (Screen 2). Swiping left on items reveals the ability to delete the item from the list. Items can be sorted by title or date when clicking on the floating menu button to reveal each corresponding action.



Screen 4

Clicking on the search menu option spawns a search box in the tool bar. The user can enter text or speak if the device is microphone enabled and results matching the entered text with the title of each verse will be displayed in the combo box. Clicking on a results item will initiate a Detail activity (Screen 2). In the event no matching items are found, the user can click enter and a full text search will be conducted on the the item's content and displayed in Screen 5.

Screen 5

Search Results Activity displays items matching a full text search. A preview of each item is shown with matching text in bold. Clicking on an item will initiate a Detail activity (Screen 2).

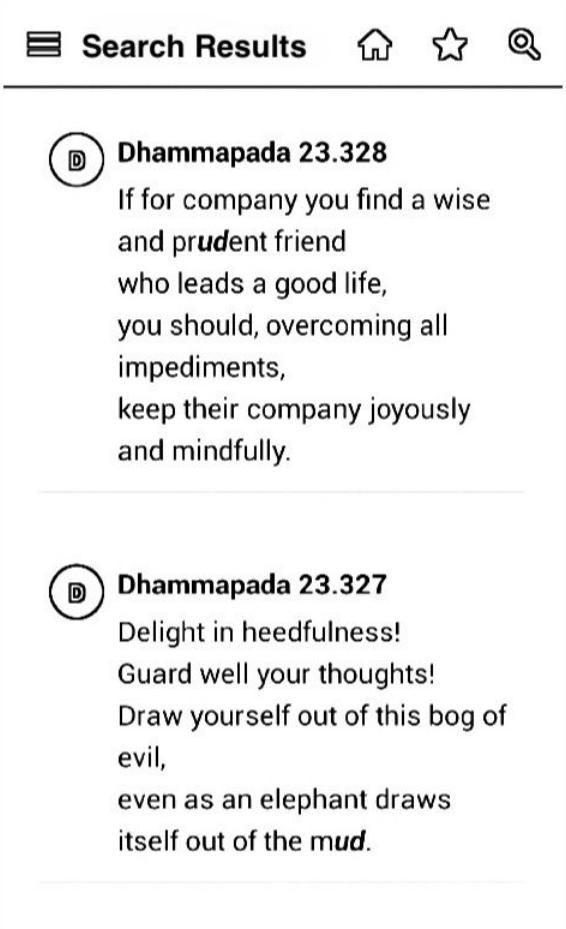
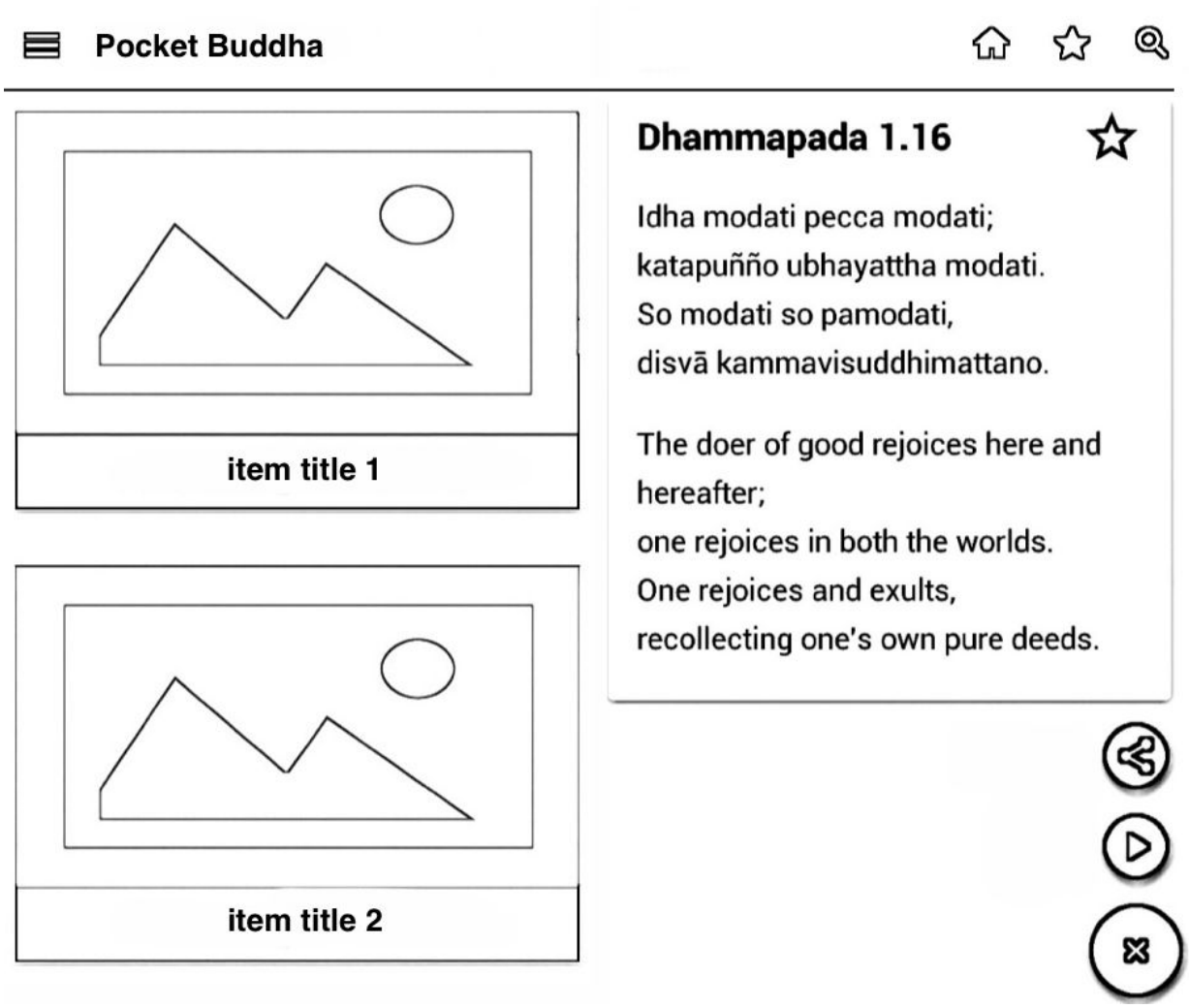


Table Layout Main Screen

Main Activity and Detail Activity in table layout.



Key Considerations

How will your app handle data persistence?

App will retrieve feed from web server endpoint in a background thread. As each entry is marked as favorite, data will persist in a local sqlite database.

Describe any corner cases in the UX.

One edge case can occur if the user decides to navigate away from an item which is currently playing audio. In that scenario, the media player can stop playback and release its resources as the view is being unloaded and the next item is shown.

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

- Glide for loading and caching images. https://github.com/bumptech/glide
- FloatingActionButton for displaying and collapsing button menus. https://github.com/Clans/FloatingActionButton
- SwipeRevealLayout for revealing delete menu of list items. https://github.com/chthai64/SwipeRevealLayout
- Material Color Library for coloring item icons http://rgr-myrg.github.io/material-color-java

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

Write out the steps you will take to setup and/or configure this project. See previous implementation guides for an example.

- Add Gradle dependencies: Glide, Swipe Reveal Layout, FAB Menu, and Material Color.
- Add server feed and search http endpoints.
- Add relevant support libraries: Cardview, Recycler View, AppCompat, Palette, Design.
- Add Google Analytics and AdMob.
- Add Internet and Network State Permissions to the manifest.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for Main Activity, Recycler View and List Item.
- Build UI for Detail Activity, Nested Scroll View and Item Card View.
- Build UI for Favorites Activity, Recycler View, Item Card View, and Swipe Reveal Layout.
- Build UI for Search Results Activity. Recycler View, and Item Card View.
- Build Menu Options with Icons: Home, Favorites, and Search actions.
- Build table layout.

Task 3: Create Main Activity

- Make http request to retrieve feed items from the server API.
- End point should specify the current page count for paging through feed content.
- Create Parcelable data object to represent feed item.
- Initialize Recycler View and Adaptor with data set retrieved from the end point.
- Use View Holder to bind data points to the view.
- Implement Glide for image loading and caching.
- Retrieve a Palette from the bitmap to match the item's title background with the loaded image.
- Set On Click listeners for items to start a Detail Activity with an Intent containing the position of the selected item.
- Implement endless scrolling.

Task 4: Create Detail Activity

- Initialize View Pager and Adapter.
- Add on page change listener for detecting the current item position.
- Retrieve the current item from the list and bind data to the view.

- Add click handlers for Sharing, Playing, and Saving items to favorites.
- Enable the floating action menu button to close when click handlers are clicked.
- Use intent to start a chooser for sharing an item. Populate with item description.

Task 5: Add Save to Favorites Functionality

- Create Sqlite helper with Database name, version, and new table for saving Favorites.
- Table columns should match Parcelable object item: title, pali, english, enclosure link, image url.
- Add a favorite column for flagging an item as favorited.
- Create Add and Remove Favorite methods.
- Create Select from Favorites table method with the ability to specify sorting order.
- Create content provider.
- Wire in on Favorite clicked handler.

Task 6: Create Sound Player

- Create class to encapsulate Media Player functionality.
- Use delegate pattern to provide callbacks for prepare, loaded, completed states.
- Expose method to load mp3 resource.
- Use async task to provide progress while loading mp3 resource.
- Expose methods to play, stop, and release player.
- Wire in on play clicked handler.

Task 7: Create Favorites Activity

- Initialize Recycler View and Adapter.
- Load in favorited items and bind data to the view in the View Holder.
- Wire in sorting capability to corresponding FAB button.
- Wire in ability to delete and item.
- Add on click listeners for starting a Detail Activity when an item is clicked.

Task 8: Create Search Activity

- Create Search Suggestion Provider.
- Use IntentService for hitting server search end point.
- Init Recycler View and Adapter.
- Use View Holder for binding data to the View.
- Handle connection and response error states gracefully.

Task 9: Create Widget

- Create Widget layout.
- Create widget class and extend app Widget Provider.
- Add widget provider metadata with update frequency.
- Use a pending intent in case the user requests a manual update.
- Add widget to the manifest.

Task 10: Create “About” Activity

- Create activity for providing information about this app and displaying content credits.
- Add credits for feed items from http://pariyatti.org and images from http://flickr feeds.

Task 11: Implement Google Ads

- Set up ad unit from the developer console.
- Pre-fetch ad by creating an instance of PublisherInterstitialAd.
- Create the ad listener.
- Request ad by building PublisherAdRequest and invoking loadAd.
- Display interstitial in ‘About’ Activity.

Task 12: Implement Google Analytics

- Create a Google Analytics Account.
- Get a configuration file from the developer console.
- Create a tracker by retrieving an instance of GoogleAnalytics.
- Track user searches and failed searches as well.
- Track app errors.

Submission Instructions

1. After you’ve completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
3. Add this document to your repo. Make sure it’s named “**Capstone_Stage1.pdf**”