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**"

Description
Intended User
Features
User Interface Mocks
    Screen 1
    Screen 2
Key Considerations
        How will your app handle data persistence?
        Describe any corner cases in the UX.
        Describe any libraries you'll be using and share your reasoning for including them.
        Describe how you will implement Google Play Services.
Next Steps: Required Tasks
    Task 1: Project Setup
    Task 2: Implement UI for Each Activity and Fragment
    Task 3: Your Next Task
    Task 4: Your Next Task
    Task 5: Your Next Task

GitHub Username: madeyedexter

Paste It!

Description

An app to save random information on a daily basis.
While browsing the web, reading a book, or interacting with any other app with visual or textual information, a user might stumble upon a piece of information they like and would like to refer

later. The app serves as a one-stop location to store all such information and browse through it later on.

Intended User

Any regular user of the digital age.

Features

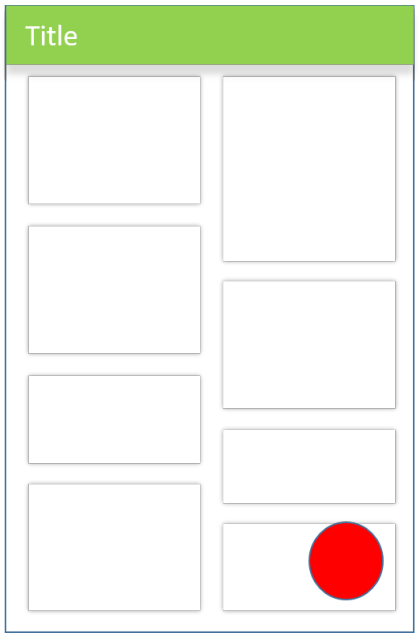
List the main features of your app. For example:

- Saves random text, pictures, url.
- Optionally notifies user based on user preference on a weekly/daily basis about the content user has saved.
- Organizes user generated content by date and tags.
- User can optionally add new tags while saving the content.
- To save a content user may share it from another app, upon which a screen will appear which will allow user to save the content, or cancel the operation.

User Interface Mocks

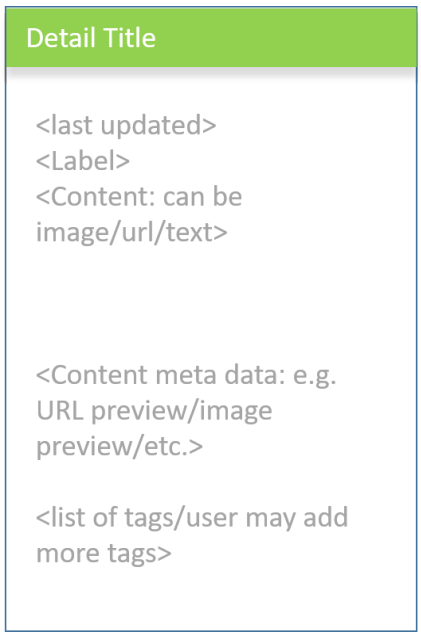
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Screen 1



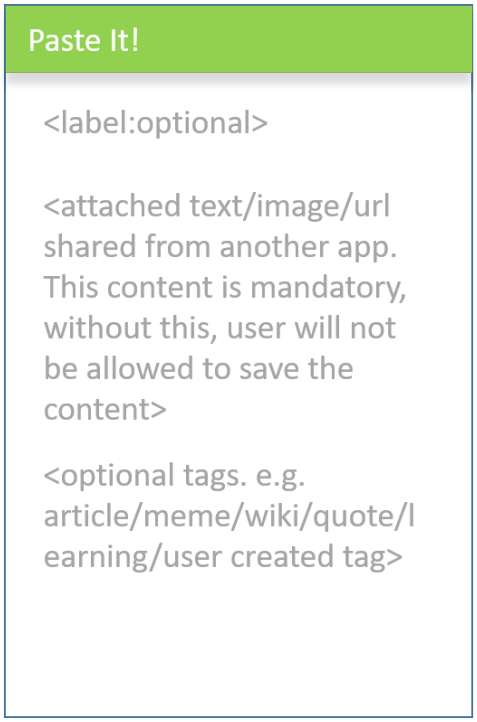
The main screen will have cards corresponding to every item user has saved till now. It will also have a FAB which will allow user to add more content from the main screen. Main screen will also have a Settings menu option which will take the user to the Settings Activity.

Screen 2



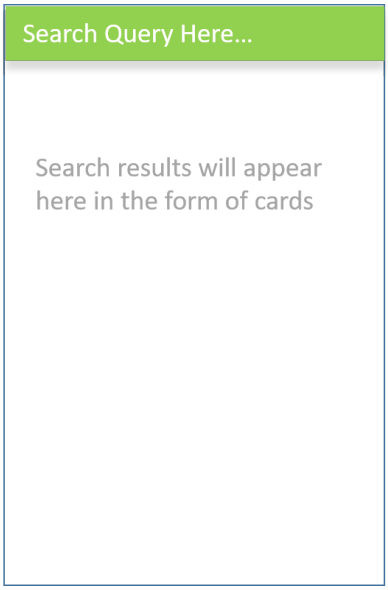
On tapping a card, the user will be shown a detail screen which will show the details of the card

Screen 3: Adding new content

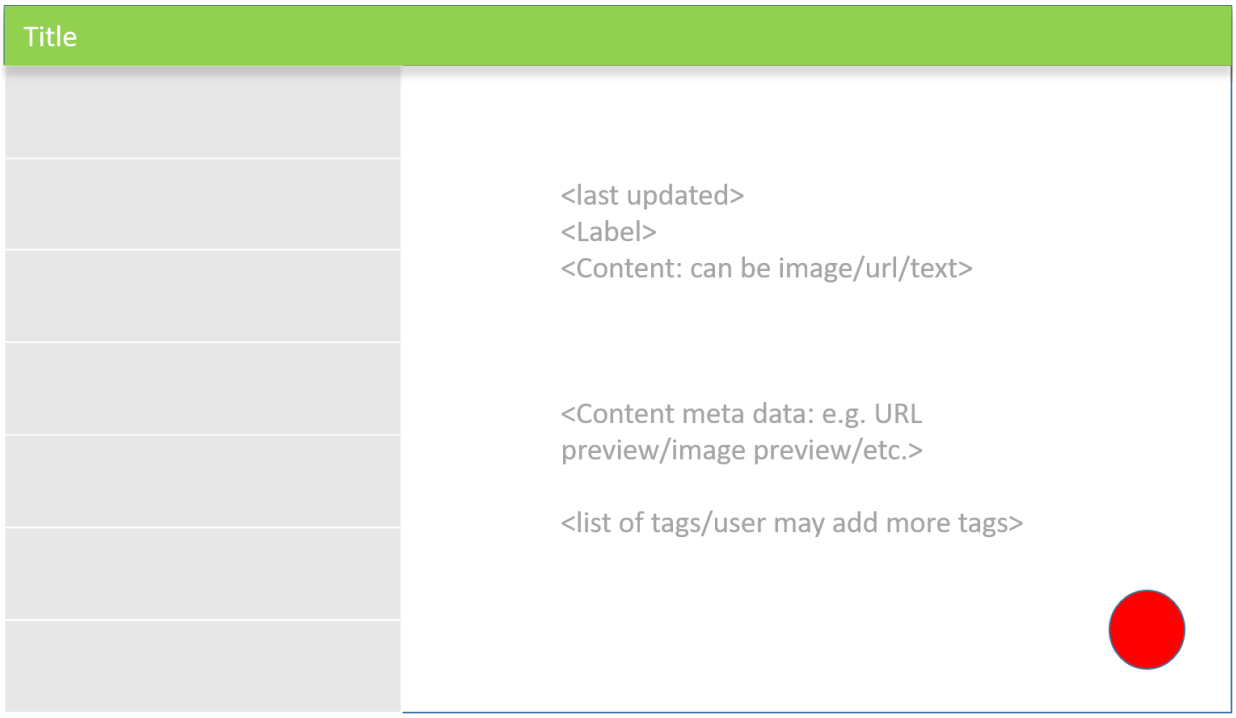


On tapping the add FAB from main screen, or by sharing an item from another app, user will trigger the launch of the add item screen. This screen will have editable components through which user may further edit items before saving them. If the user launches this screen from Main Screen, they will be returned to the main screen. If the user launches this screen from another app, they will be returned to the app they were using when this activity is finished.

Screen 4: Search Screen



Screen 5: Tablet UI



Screen 6: Widget

Widget will contain last 5 saves by the user in the form of ListView. Minimum size 2x2

Label content
Label content
Label content
Label content
Label content

Key Considerations

How will your app handle data persistence?

- App will persist data to the Firebase real time data base.
- App will also maintain a cache of the data locally using greendao orm backed by SQLite.
- If a user saves a URL as part of their content, the open graph and related metadata for the urls will be fetched and persisted locally for user preview. The meta-data fetch and cache operation will be done using an intentservice.
- App will use Picasso's caching system to fetch and cache images.
- Since firebase has an asynchronous architecture, we will not need any other explicit asynchronous calls.

Describe any corner cases in the UX.

Described in screen 3 above.

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

- Picasso for loading images
- Firebase RTD for syncing user data.
- Firebase auth for creating user accounts
- Firebase storage for storing images.
- Greendao for caching data in SQLite
- To fetch open graph info: https://github.com/johndeverall/opengraph-java

Describe how you will implement Google Play Services.

- Google Play Auth for authentication: App will maintain users sign in information. If user is not signed in, they will be shown a sign in activity before using they can use the app.
- Firebase: All data will be fetched from sqlite database using loaders and greendao. Firebase listeners will listen for data changes and update the local database using greendao api calls (insert/update/delete/etc.). If the user is currently viewing a particular item the change will be made in the Firebase completion callback to reflect changes in the UI.

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 sub-tasks:

- Add libraries
- Configure app for firebase
- Setup Manifest
- Add Models

If it helps, imagine you are describing these tasks to a friend who wants to follow along and build this app with you.

Task 2: Implement UI for Each Activity and Fragment

List the subtasks. For example:

- Build UI for LoginActivity
- Build UI for Main Fragment
- Build UI for Add content Activity/Fragment
- Build UI for Detail Fragment
- Build UI for search fragment

Task 3: Read Operations

Describe the next task. List the subtasks. For example:

- Write sign-up/auth code
- Implement Loader Callbacks in main fragment

- Implement Loader callbacks in search fragment

Task 4: Write Operations

Describe the next task. List the subtasks. For example:

- Write code for persisting notes
- Write code for firebase completion callbacks
- Write Service/Broadcast receiver to remind user of the data they have saved.

Task 5: Widget

Describe the next task. List the subtasks. For example:

- Write a widget which displays the last n items. Read n from User Settings/Preference.

Task 6: Optimize UI and Polish

- Polish app according to material design spec.
- Optimize layout for various screen sizes.
- Externalize string/dim/bool resources

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**"