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: madza1812

NoteYou+

Description

Problem: When users have some light-bulb ideas on train, some todo-list from works, or some forgot memos which supposed to be on the fridge's doors, they do need to note down right away otherwise, those ideas, memos and notes might be forgotten during the day.
Solution: NoteYou+ app is simply designed to capture users' ideas, thought, memos, checklist and save them securely and automatically in the device or in the cloud, and also help them to

share the note efficiently rather than put them on the fridge door which might be not be noticed accidentally. Indeed, NoteYou+ is a simple and lightweight notepad application not only does that allow you to capture your ideas quickly and securely, but also does that allow you to organize them based on their natures and share them with family, mates, and colleagues.

Intended User

NoteYou+ app can be used by everybody.

Features

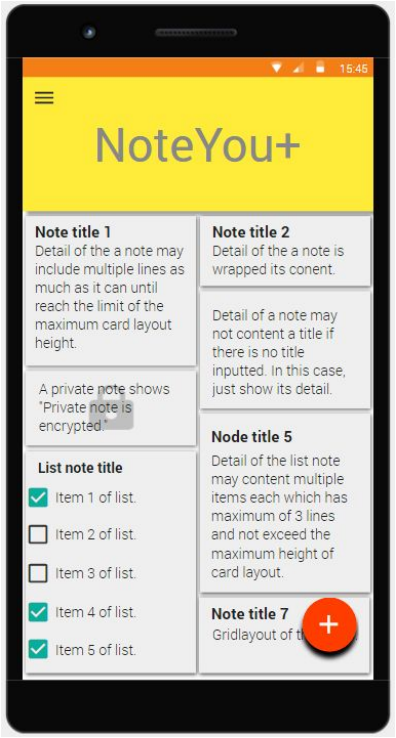
Main features of NoteYou+ app:

- Take notes (text) or todo-list (checklist).
- Automatically saving.
- Cloud backup ability when saving notes with an account.
- Securely with Encryption when needed.
- On only sharing the content of 1 specific note, but also sharing a specific note with other NoteYou+ user, who will be able to edit that shared note.

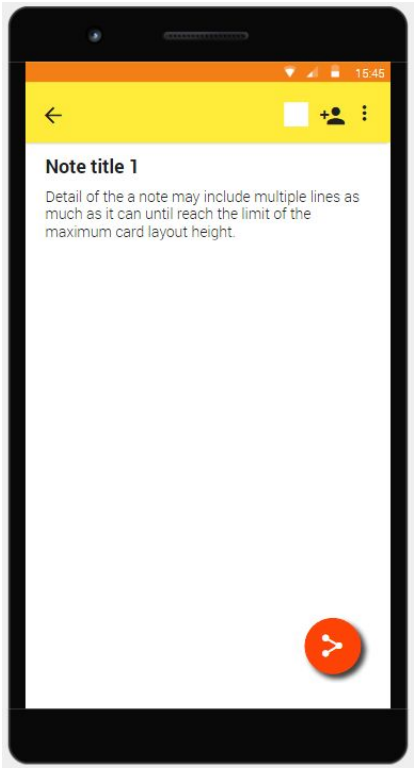
User Interface Mocks

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.

Main screen: showing all the notes when launching the app.

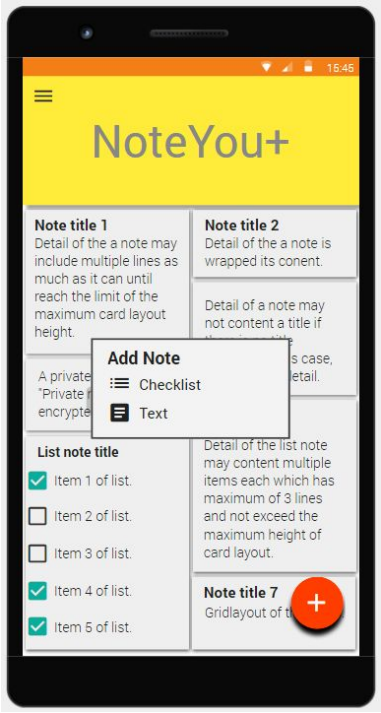


Detail text note: editable text note screen.

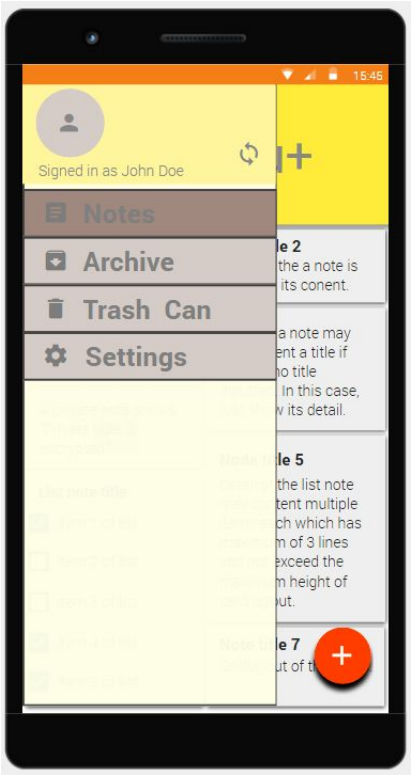


Detail todo-list note: editable todo-list note screen.

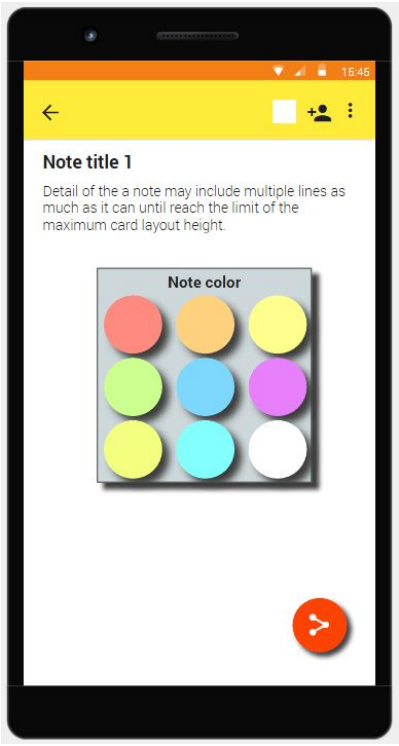
Adding new note screen: shown when click the FAB plus button on main screen.



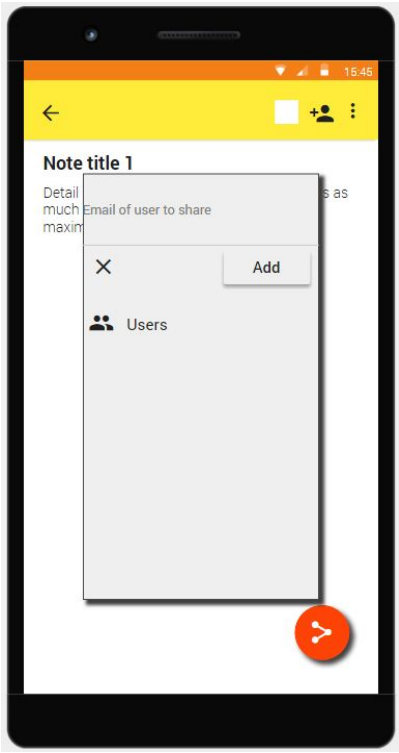
Drawer screen: shown when user click the drawer icon on the top-left screen.



Color picker screen: shown when user tap color icon when editing a note.



Sharing note screen: when user tap on the share icon on the toolbar (not the FAB Share button) when editing a note.



Key Considerations

How will your app handle data persistence?

Database is handle by SQLite with multiple tables includes:

- 1 notes_table capturing note_id (int, primary, auto increase), notes (text), note_type (int, not null), created_user (varchar, not null), created_date (date, not null), modified_user (varchar, nullable), modified_date(date, nullable), encryption_pass(varchar, nullable), color (int, not null), etc.
- 1 users_table capturing user_id (int, primary, auto increase), user_firstName (varchar, not null), user_lastName(varchar, not null), user_pwd (varchar, not null), created_date(date, not null), etc.
- 1 sharing_table capturing share_id (int, primary, auto increase), note_id (int, not null), user_id (int, not null), etc.

Columns number may vary during designing the app if it's necessary.
This database system will be used for the SQLite in the local device and SQL database for the cloud system based on google app engine.
In local device, the content provider will be built upon this database. When the app is used with the cloud setting on, the notes will be fetched from the cloud, and the sqlite will save all the table. The data synchronization between the cloud and local devices will be based on the sync provider or custom synchronization using service and HTTP post to send data to Backend API, implemented by Google App Engine.

Describe any corner cases in the UX.

Click Back button on Main screen will exit the app. The app can be navigated by the drawer navigation and will usually is used in 2 screens, Main Screen and Detail screens. Other screens will be used as additional dialogs adding upon the previous screen for some specific user interactions such as color picker, encryption, etc.

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

- Picasso for network images loading.
- ButterKnife for viewholder.
- Volley for network JSON requests.

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

- Create new Android project with Android Studio.

- Create new Backend API Endpoint module in Android Studio.
- Configure build.gradle to include into the app project the backend module, picasso lib, volley lib and other libs as needed.
- Configure GCM moudle to pushing message to new shared SQLite.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity, Main Fragment using Gridcard layout.
- Build UI for Card Items.
- Build UI for Activity Detail.
- Build UI for the Fragments of the Text Note and the Todo-List Note.
- Build UI layout for the color picker, add shared users, and create new dialogs.
- Build UI layout for the Drawer Navigation.
- Build Setting UI.

Task 3: Implement Data Persistence with Content Provider

Create tables as described in Data Persistent part above in SQLite:

- Implement SQLite database includes DbContract, DbHelper, etc. And test it.
- Build a content provider based on the created database and test it.

Task 4: Implement the Google App Engine.

Data from local is pushed to the backend module through Http Post. Then Google App Engine (GAE) will persist the received data into its database system, Google datastore. Request from Android Client based on the endpoint api is responded with data queried from Google datastore:

- Create tables for Google datastore.
- Implementing the data persistent into the Google datastore from Http post request.
- Implement the Endpoint API with data are queried from Google Datastore and packed in the JSON type.
- Testing thoroughly.

Task 5: Implement the service for loading notes from GAE backend module.

The Intent service to load the notes from Google App Engine module and save the notes into the local SQLite database.

- Create a new Intent Service
- Loading the notes from Backend Module and insert new values into the SQLite database.
- Handling no internet connection,network error, server error, etc.

Task 6: GCM module

When a not is shared, a notification is push through the GCM to the shared users.

- Configure the GCM module.
- Implement the notification to launch the detail activity of the shared note.

Task 7: Configure the free and paid version in Gradle

- Configure build variant in the build.gradle script.
- Implement free version includes Google Ads banners at the bottom of the main and detail screen, also the full screen Ads when change from Detail screen to Main screen, disable the encryption in free version.

Task 8: Tablet, widget, and wearable

- Build a tablet UI with the dialog activity for the detail screens by change the Detail Activity to realize the tablet screen to show as full screen activity or dialog activity.
- Build UI for a widget to show list of notes (3x3).
- Implement the widget so that when user tap on a note in widget to launch the detail activity of the tapped note.
- Build a notification for wearable.

Task 9: User testing.

Use the application with all the feature for couple of days.

- Testing thoroughly the application.

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**"