GitHub Username: shivaq

Formerly

Description

"Formerly" is a game and a kind of diary. You can tap and record your emotions quickly. And these emotional taps are counted and which changes appearances of characters.
Basic elements to record are Who feels, How feels, about What.
For instance, You feel joy and surprise about a meal. A samurai fears about the pain of his wound.
Your records and statistics are securely protected with a password when you read them.
Of course, you can add details about emotion events lately.

★ Tutorials
We provide tutorials. What do you think how a samurai, an animal, a tree feels about something?

★ Avatars
Characters have its avatar. Avatar assigned to them will change in some condition.

★ Dashboard

You can see frequency and relations of emotions and characters.

Intended User

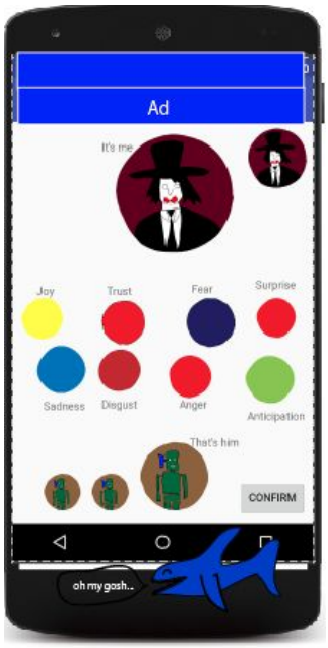
People who want to know more about her/his own emotions.

Features

- Record your emotions quickly.
- Protect your private data with a pin code.
- Characters change its appearance according to your emotions and relations.
- You can see which character arouse your emotions most.

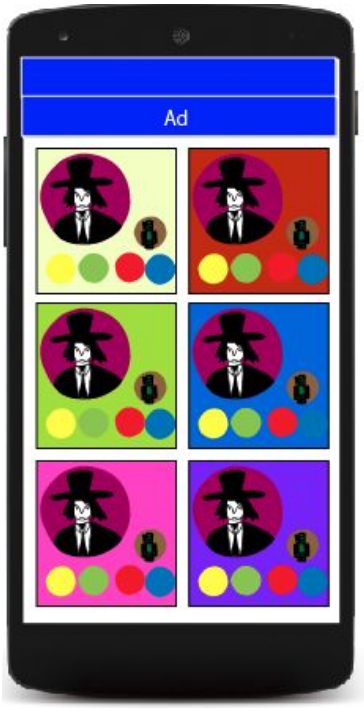
User Interface Mocks

Screen 1: MainRecordActivity



Activity to record users emotions.
A user selects who feels what emotions about what.
When the user set and confirm them, the assistant appears and suggest next action.
The user can choose to write in detail about the emotion event or go to emotion event list or add a new event.
The user can wipe out the assistant with swipe action.

Screen 2:EventListActivity



Activity which shows emotion events with recyclerview.
When the user tap an item, intent to the DetailActivity

Screen 3:DetailActivity

Activity to write about emotion event in detail.

Users can add:
- The specific name for the characters.
- The role of the characters at the event like a father, student, customer, or nobody.
- Explanatory notes.
- Relevant events.
- Change event date.

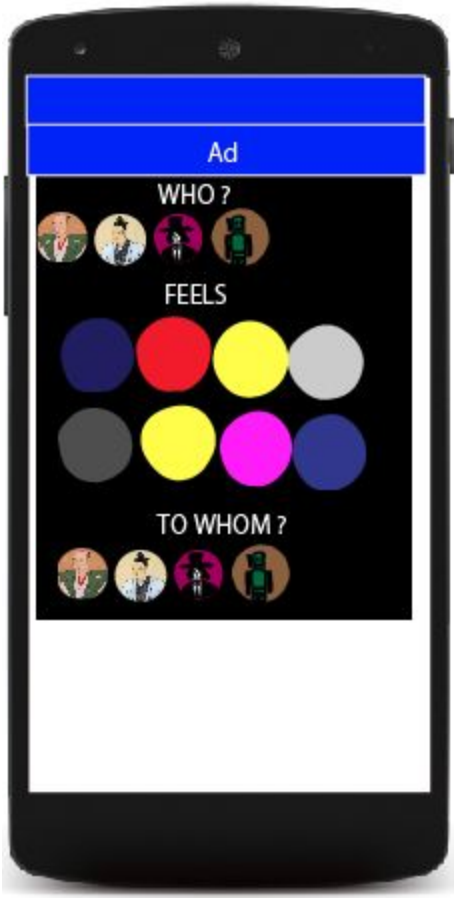
Screen 4:TutorialActivity

Provides emotional events examples.
Users can learn how to use this app.

Screen 5:DashboardActivity

Shows the ranking like the most joyful characters.
Shows emotional days of users history.

Screen 6:Widget

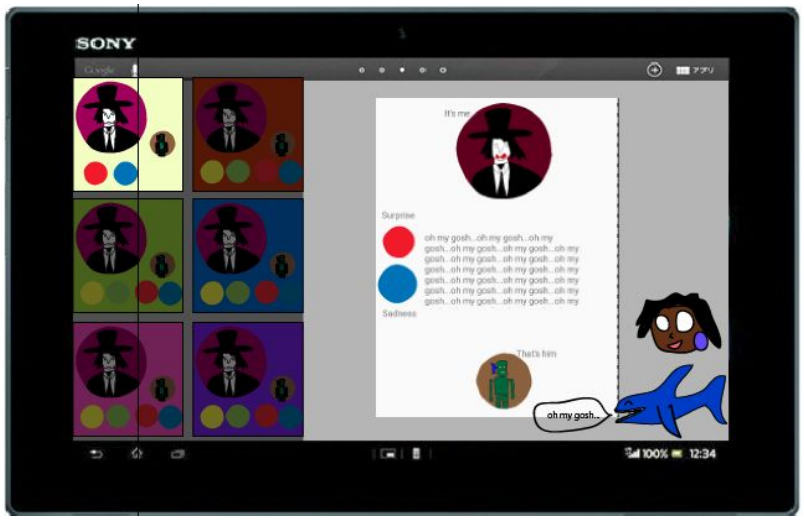


Tap and record users emotions.

Screen 7:SettingsActivity

Set pin code for entering Activities except for MainRecordActivity.

Screen 8:Tablet



Provides 2 pain mode.

Key Considerations

How will your app handle data persistence?

Store data with Content Provider.
Retrieve local data with AsyncTaskLoader.
Fetch some images from Google Cloud Endpoints.

Describe any corner cases in the UX.

Internet Connectivity :
If there is no network while the app fetches image from Google Cloud Endpoints, use a local image.

Error handling for AsyncTaskLoader:
Create a Wrapper AsyncTaskResult<T> that holds a generic and an exception. And let it handle errors.

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

dagger2: For making architecture loosely coupled.
Google Cloud Endpoints: To store images and fetch them.
Picasso: To fetch data from Google Cloud Endpoints.
butterknife:To increase readability of code.
timber:For easy logging.
assertj:For testing.
mockito:For testing.
robolectric:For testing.
espresso:For testing.
stetho:To check SQLite easily.
support-vector-drawable:To use SVG.
MPAndroidChart:Generate chart for emotion statistics.

Describe how you will implement Google Play Services.

Analytics:To analyze users.
Notification:To tell user new features or new events.
AdMob:For ad.

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 directory structure for MVP pattern.
- Configure basic functions for dagger2.
- Configure gradle tasks.

Task 2: Configure firebase services

- Configure Admob.
- Configure Analytics.

Task 3: Implement UI for Each Activity and Fragment

- Build UI for MainRecordActivity.
- Build UI for EventListActivity.
- Build UI for DetailActivity.

Task 4: Configure Google Cloud Endpoints

- Configure Google Cloud Endpoints.
- Configure java library for changing the image without updating the app.

Task 5:Implement logic for MainRecordActivity

- Configure Insert operation for emotion data.
- Store app launch counter to sharedPreferences.
- Configure dialog to inform image update.
- Configure drawer.
- Intent to EventListActivity or DetailActivity or TutorialActivity.

Task 5:Implement logic for SettingsActivity

- Configure pin code setup logic.
- Configure sort order preference.

Task 4: Configure databases

- Configure databaseHelper.
- Configure Content Provider.
- Configure AsyncTaskLoader.

Task 5:Implement logic for ListAdapter

- Fetch images from Google Cloud Endpoint and store them to SQLite.
- Load data from local to EventListActivity.

Task 5:Implement logic for EventListActivity

- Configure FAB to change sort order.
- Use RecyclerView to show event list.

Task 5:Implement logic for DetailActivity

- Fetch images from Google Cloud Endpoint and store them to SQLite.
- Add detail data to emotion event.

Task 5:Implement logic for DashboardActivity

- Configure MPAndroidChart.
- Query data to show some rankings.

Task 5:Implement logic for TutorialActivity

- Shows tutorial list with RecyclerView.

Task 5:Implement Widget

- Build UI for a widget.
- Implement logic intent from widget to DetailActivity.

Task 5:Implement logic for notification

- Configure notification to notify availability of new character to users.

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**"