GitHub Username: SurrealTiggi

Reward Bingo

Description

Reward Bingo is an app to provide easy chore tracking for parents to reward their children. Instead of buying disposable reward charts and stickers, this app allows parents to build their own charts on the fly and keep a historic record of their child's progress.

Intended User

Reward Bingo is built for parents, but children from a certain age can use it to build their own charts as they wish. There is no limit as to how many children or charts are tracked and kept.

Features

Reward Bingo allows you to do things like:

- Create weekly charts
- Choose from a few pre-built charts, or re-use older ones
- Share rewards on other social media
- View historical charts and summaries

User Interface Mocks

Main activity

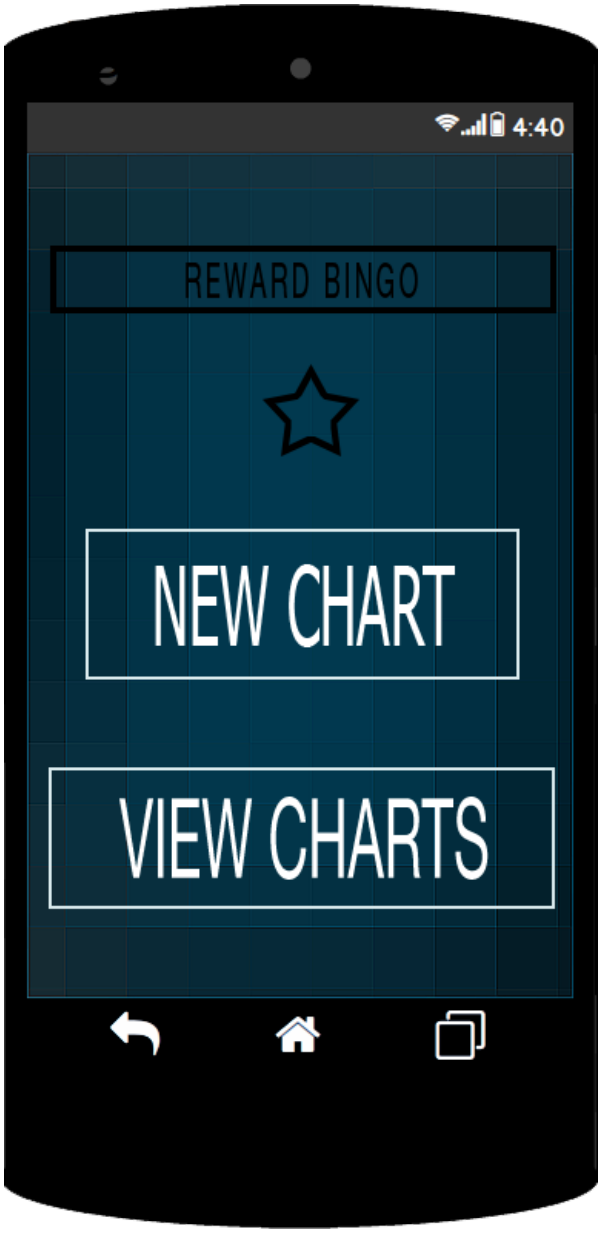
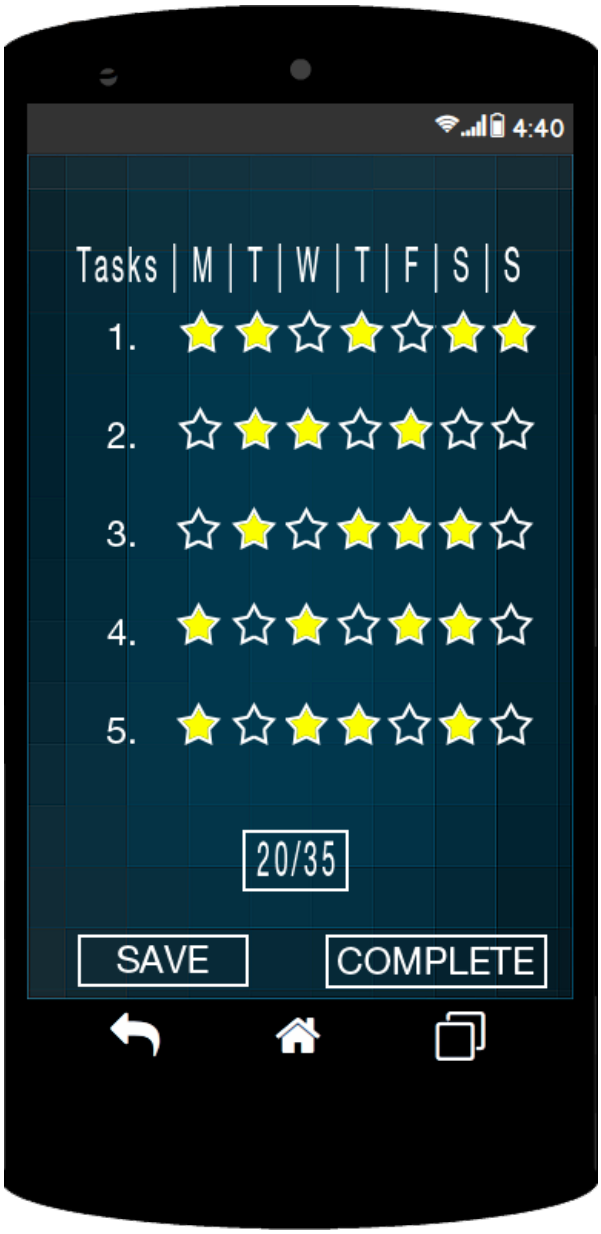


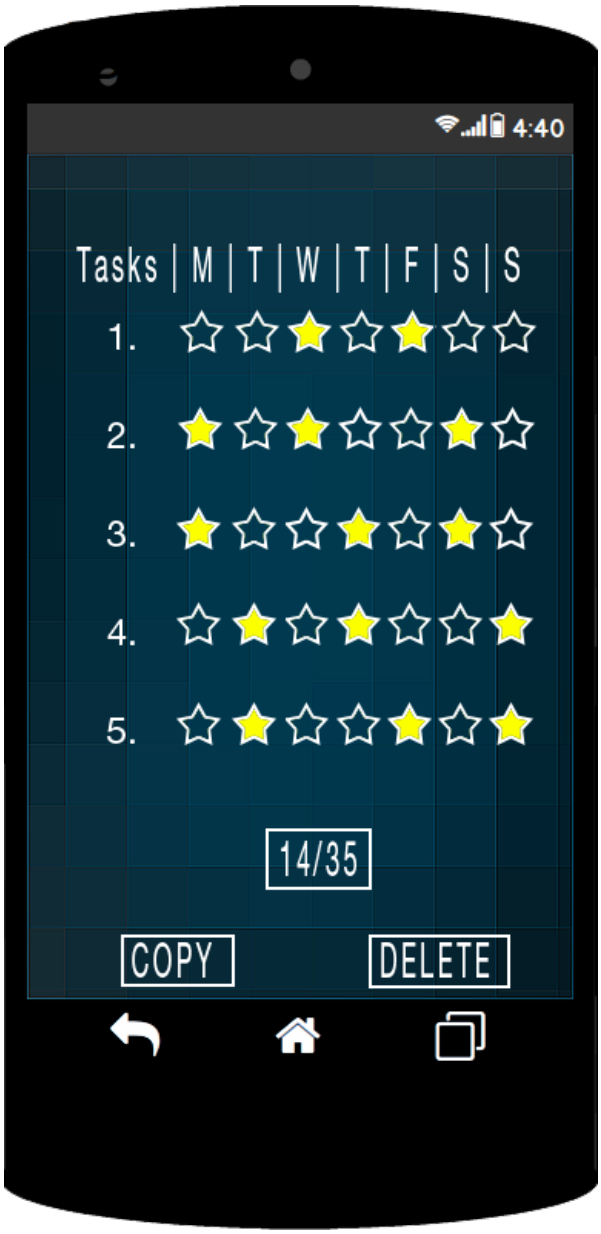
Chart creation screen

Chart view screen

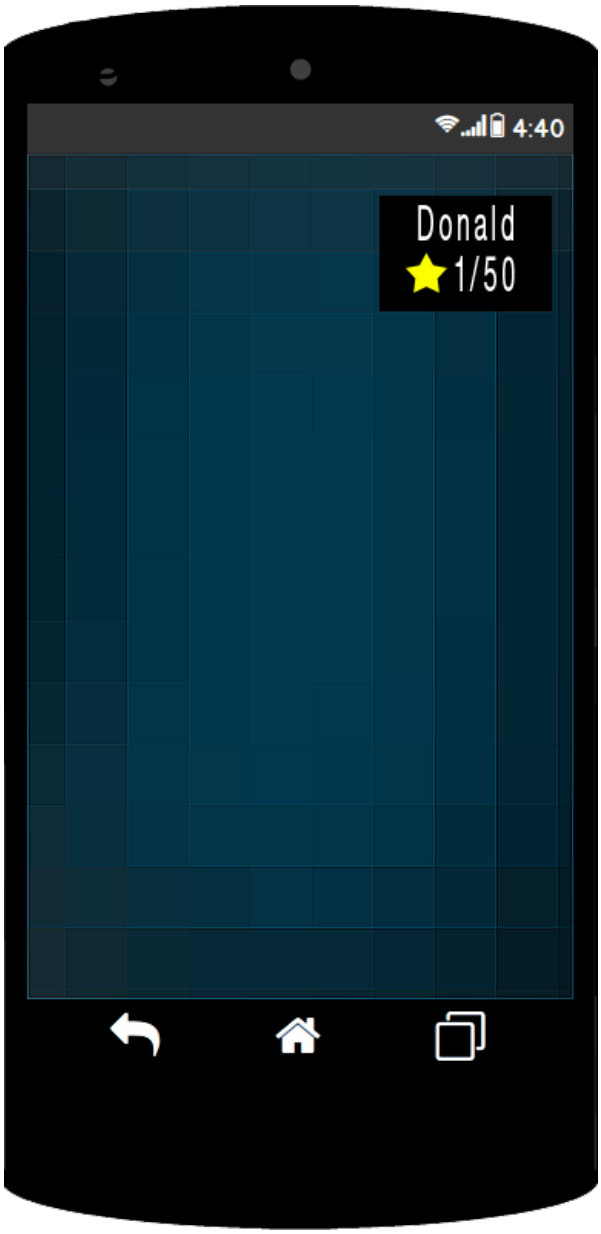


History screen

History view screen



Widget



Key Considerations

How will your app handle data persistence?

The app will store all data locally, using a SQLite database and a ContentProvider to work with this data.
Pre-built tasks will be stored on a web-based API and fetched using an IntentService.

Describe any corner cases in the UX.

1) When creating a chart:
    a) The back button will be disabled, and only buttons to return to the main screen or to complete the chart (thereby saving it locally), will be available. If the user clicks to go to the main screen, they will be warned that the chart will disappear which they must confirm. A user must also be specified or completion will be disallowed.
    b) If the user does anything to call onPause() (eg. press the Home button, or switch off the screen), the app will persist the chart object, and restore it when the user returns to the app, but it will not save it locally unless the user selects to save the chart.
    c) IntentService will only be called when “New Chart” is clicked from the main screen and loaded into a local array.
    d) Tasks will be limited from a minimum of 2 to a maximum of 7.
    e) If New Chart is clicked when there is an existing chart that hasn’t been completed, the user will be prompted on creation that the older chart will auto-complete and be saved into history.
2) When viewing an existing chart:
    a) The back button will be enabled, and if pressed will save the chart without prompting.
    b) When the user presses “Complete Chart”, they will be prompted to confirm in case it was accidentally pressed.
    c) Due to screen space limits, tasks will show numerically, but expand into an info popup when clicked.
3) When viewing history:

a) Pressing delete will ask for confirmation.
b) If only there's only one chart and one user, the user will also be removed.
c) Chart editing will be disabled on this activity.
d) Copy will load tasks into an array, and start a new "New Chart" activity
e) History screen will only show charts that have been fully completed, not in-progress charts.

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

ButterKnife for easier variable management and view injections.
OKHttp for in-line ASync tasks integration in IntentService to keep services contained and easier to manage.

Next Steps: Required Tasks

Task 1: Project Setup

First step is to make sure we can get started.

Subtasks:

- Configure libraries
- Configure gradle
- Implement basic colors and styles in relevant xml

Task 2: Implement UI for Each Activity and Fragment

Now we can start designing each screen.

Subtasks:

- Build UI for MainActivity
- Build UI for New Chart
- Build UI for View Chart
- Build UI for History Chart
    - Build UI for History View (fragment)
- Build Widget layout

Task 3: Implement saving and retrieving charts from local DB and API

Next we need data to be stored/retrieved locally via the creation activity.

Subtasks:

- Implement basic SQL class and ContentProvider
- Implement cloud based API to fetch random tasks
- Implement IntentService to read from above API

Task 4: Finalize Main Activity into New screen

Now that saving works, we can align tasks better.

Subtasks:

- If parent is Main Activity, run single IntentService to fetch from API
- If parent is History View, don't run IntentService

Task 5: Implement view/summary screens

Should just be summary screens left now.

Subtasks:

- Design main view screen to allow real-time tagging of done/not-done tasks.
- Implement correct functionality to save and complete single chart.
- Draw basic history summary screen
    - Extend to fragment to show read-only single chart, with options to copy and delete
    - Copy reverts to main activity with tasks now pre-populated.

Task 6: Misc leftovers

Most functionality should be working, now we can do leftovers.

Subtasks:

- Implement simple widget that shows latest user chart and numerical summary
- Implement FAB to share View Screen
- Implement Admob on main activity only
- Implement Analytics