
GitHub Username: cecyurbina

List now! Simple share list

Description

"List now" allows users share a list of supplies quickly and easily in a real time list application. It can be used as an assistant in home grocery shopping and in special events among the members involved.

Intended User

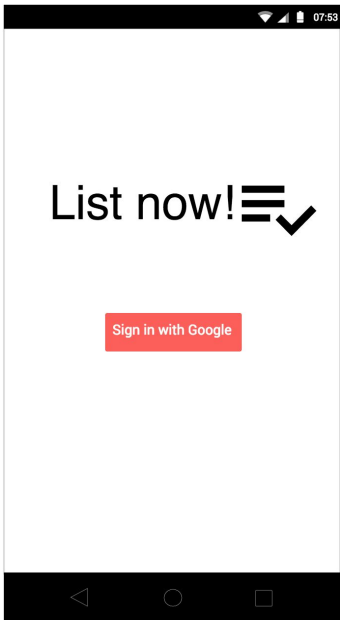
For all those who need to share their purchases or organize the supplies of an event with their family or friends.

Features

- Create lists
- Share lists among more people
- Modify lists in real time
- Upload images
- Accept or decline products added on list
- Login with google accounts
- Widget with favorite list

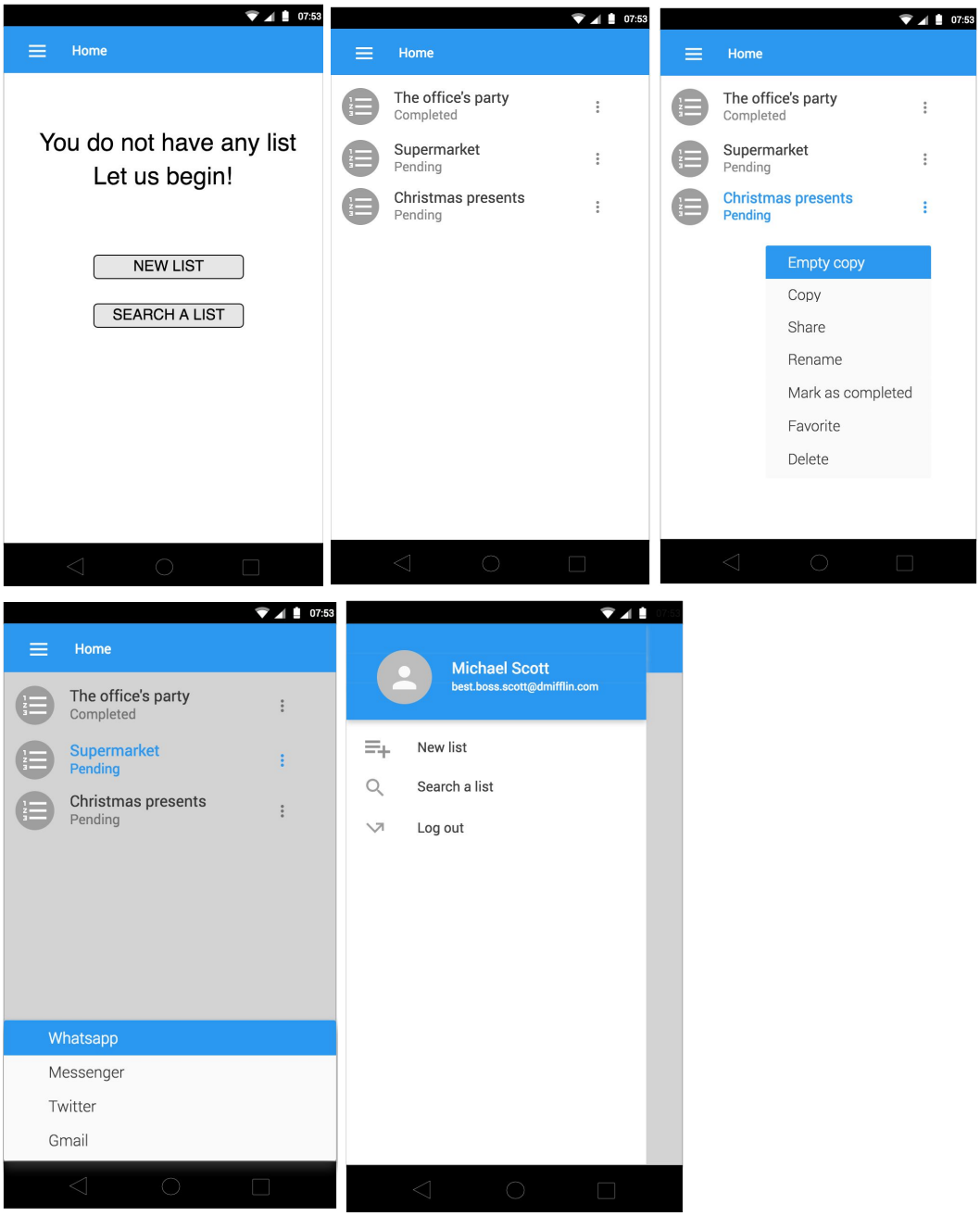
User Interface Mocks

Screen 1: Login



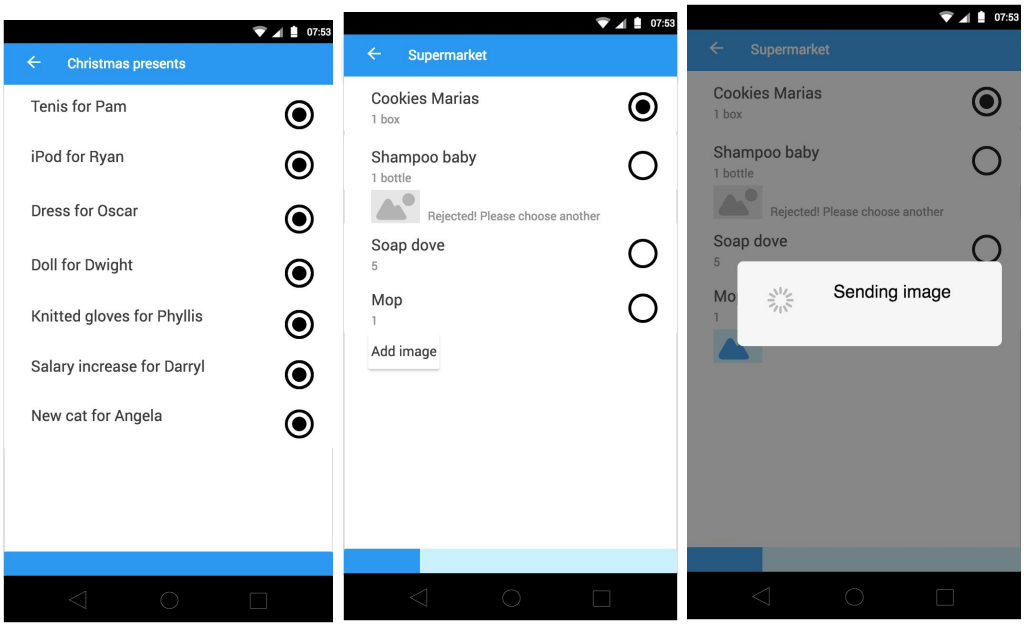
Signup and login with google account

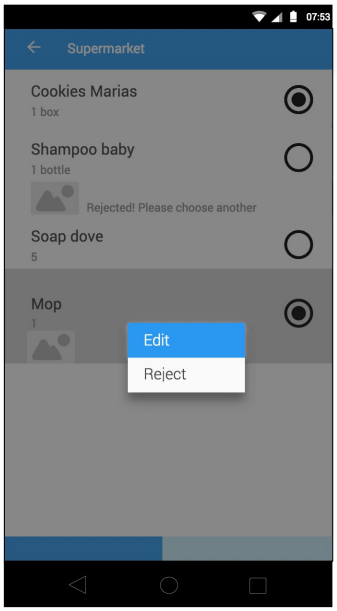
Screen 2: Home



Here the app shows the lists that the user has and all the actions that every list have. For example the user can share the list id using some message apps for eventually the other user can search the list. The favorite option marks a list to show in the widget. If the list is empty then the view will show a message for the user to start having lists. This view also have a navigation drawer with the main actions in the app.

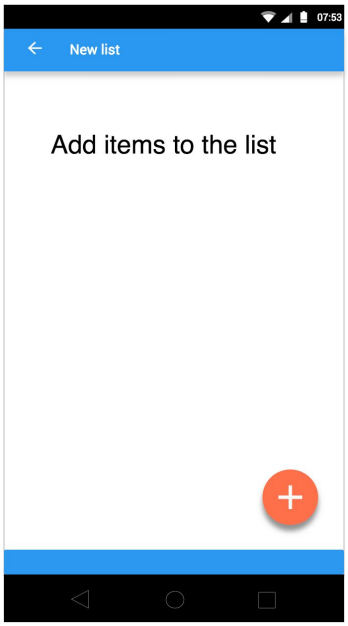
Screen 3: Filling the list





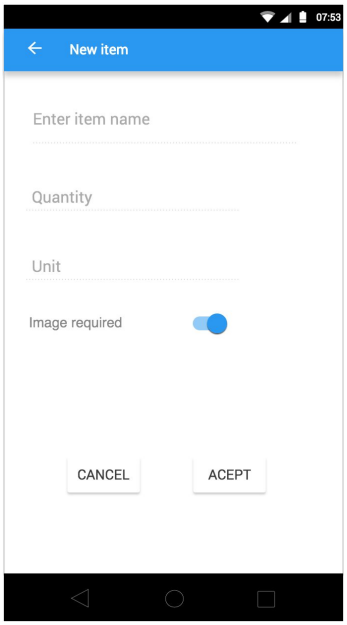
Here the user can start filling the list and users who have the shared list can also fill or reject any product filled. The list's items can have a photo according to the configuration of the element. At the bottom, the level of completion of the list is shown.

Screen 4: New list



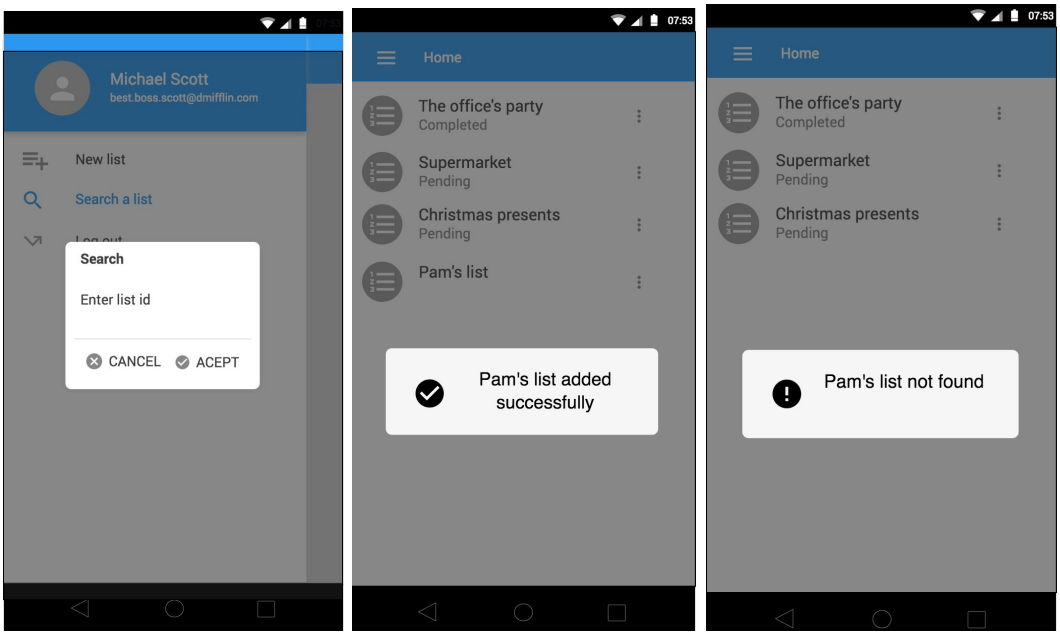
Here the user creates a new list and adds elements

Screen 5: Add/Edit item list



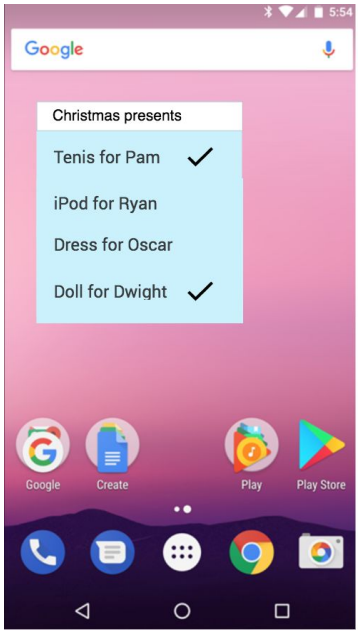
This screen allows create or edit a list's element where item name is the only required element. The required input will have an indicator like "*".

Screen 6: Search list



This action allows the user search a list by id.

Screen 7: Widget



The widget will show the last list marked as favourite.

Key Considerations

How will your app handle data persistence?

Data will be store with Firebase Realtime Database to allow share data between users.

Describe any edge or corner cases in the UX.

- Connection errors (offline or weak connection): app will not crash in that cases.
- Rotation screen: the app will restore the view states without any crash.
- List removed: when user A (owner) removes a list and there user B using it, the app will handle that case sending a message to the user B warning that the list has been deleted.
- Large images: the app will make a resizing.

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

The application will be written in Java language only and with Android Studio 3.1.2 and Gradle 4.4

- Glide 4.7.1
    - Glide will be used for simplify the loading of remote images
- Firebase
    - For store data, upload images and authentication
    - Firebase authentication 4.0.0
    - Firebase database 16.0.1
    - Firebase storage 16.0.1
- Butter knife 8.8.1
    - For simplify code of views and events

Describe how you will implement Google Play Services or other external services.

The application will use **Firebase Real data time** to save the shared list of the users also **Firebase SDK Authentication** to authenticate with google account and **Firebase storage** to save images for elements of the lists that require it.

Next Steps: Required Tasks

Task 1: Project Setup

- Create an android project in Java language only.
- Add the necessary libraries.
- Add firebase configurations for all services the project requires.
- Add theme and main colors in styles.
- Add navigation drawer with actions.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for Home (Activity and menus)
- Build UI for New List
- Build UI for New item
- Build UI for fill the list
- Build UI for Search (Dialog)
- Add all strings for views on strings.xml
- The app enables RTL layout switching on all layouts
- The app includes support for accessibility. That includes content descriptions, navigation using a D-pad.

Task 3: Login

- Integrate login functionality with firebase and google account.
- Build UI for login

Task 4: Create list

Implement Firebase Realtime Database.The app performs a short duration, on-demand requests the app uses a class like AsyncTask but Firebase database has its own async task with Database reference and Listeners also firebase storage has Upload task instead Asynctask.

- New list: Integrate UI with functionality using firebase realtime database
    - Write on Firebase database the json with a standard structure like.

```
{
  "name": "Christmas party"
  "id":1
  "owner": {
    "ownerId":"11223"
    "ownerName":"Michael Scott"
  }
  "completed": false
  "items": [
      "name":"Tennis for pam"
      "image":true
      "unit":null
      "quantity":1
    ]

  }
}
```

    This structure is just an approximation it can change if I need that.
- New item: Integrate UI with functionality using firebase realtime database
    - Write every item in the Firebase database.
- Add validations

Task 5: Functionality to “Fill the list”

- Check elements
- Add images with firebase storage
    - Upload images from local storage
    - Use Upload task to upload images to Firebase storage
- Add progress bar at bottom of the view
- Listen and update view in real time
    - Write on Firebase database the list’s updates.

Task 5: Home

Add functionality to UI

- Share, rename, delete lists.
    - Get id list consulting Firebase database.
    - Rewrite the name for the list consulting by list id and then rewrite the name in the Firebase database.
    - Delete the list with a connection with Firebase database.
- Add the share option

Task 6: Search

Add functionality to UI

- Search lists by id
    - Get the list with the id provided consulting Firebase database.
- Add list in home
- Error and success messages

Task 7: Widget

Add widget and the favorite functionality in lists

Task 8: Auto review

- Check everything is complete about functionality.
- Support rotations and garbage collector without crashing.
- Catch all exceptions

Task 9: Unit test

Add espresso tests for automated UI testing.