- Description
- Intended User
- Features
- User Interface Mocks
  - Screen 1
  - Screen 2
  - Screen 3
  - Screen 4
  - Screen 5
  - Screen 6
- 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: Google Play Services
  - Task 4: Build Flavors
  - Task 5: Polish UI
  - Task 6: Create Home Screen Widget
  - Task 7: Build App

GitHub Username: Jforonda

Split Check

Description

Split Check is an Android Application designed to help users split checks or bills in an easy and organized fashion. This app will ease the pain of having to calculate how much each person in a group owes when going out to eat, or paying any kind of bill. Next time your group gets a check, instead of pulling out your calculator app, pull out your Split Check app!

Intended User

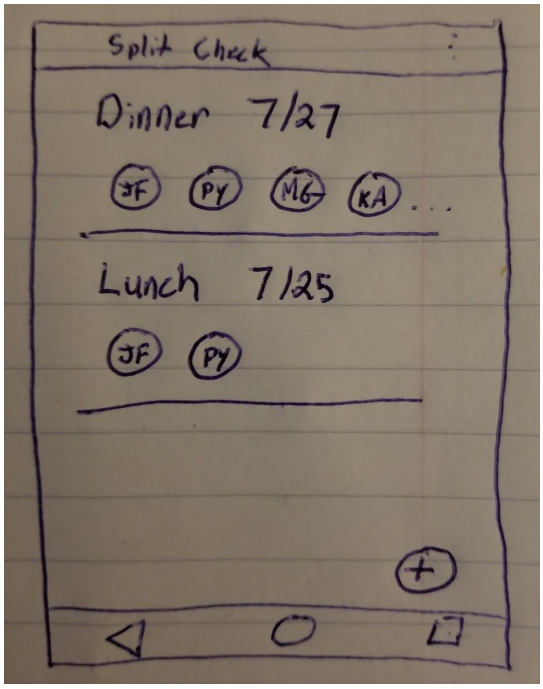
The intended user for this application is anyone who likes to go out to eat. For the most part, this will target students who like to go out to eat with others.

Features

- Calculates how much each person in a group owes toward a bill
- Saves transactions to track old bills and see if any money is still owed
- Provides a breakdown of what each person is paying for
- Handles all cases found when paying a check (tax, tip, gratuity, card minimum)
- Has a home screen widget which shows money which is owed

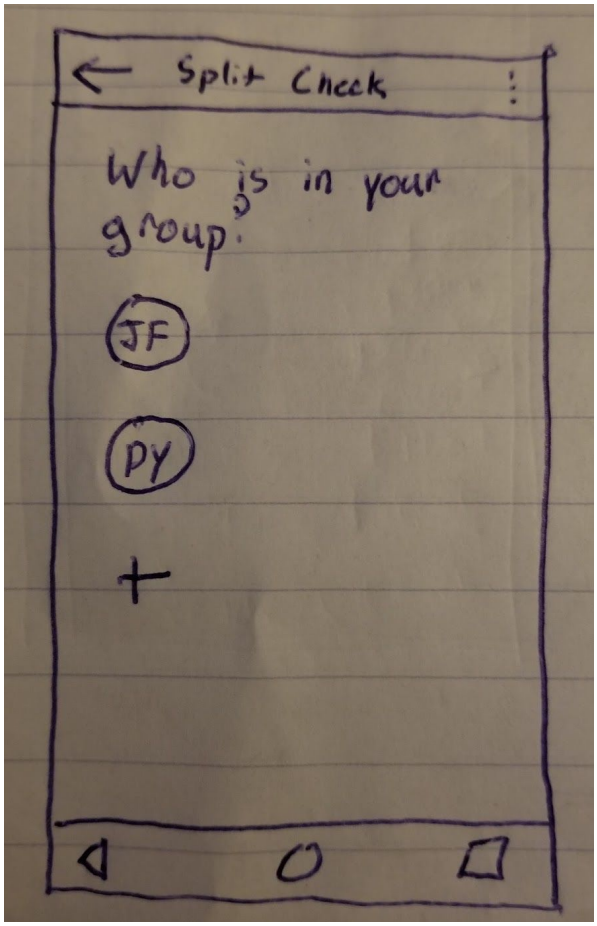
User Interface Mocks

Screen 1



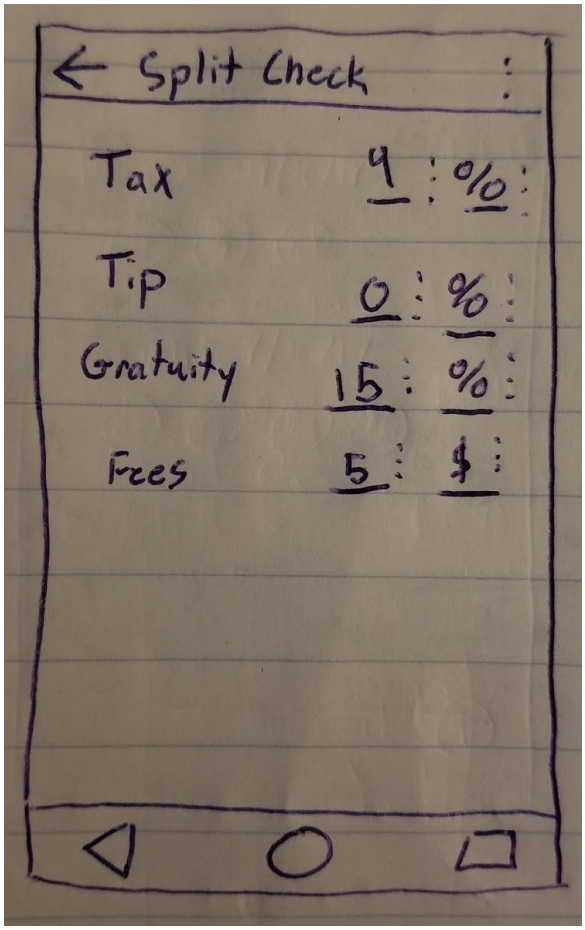
This will be the main activity which will show the current and pass checks that have been split. This will also be where the user can create a new check to split using the FAB. From this screen the user can determine from a glance, which check they have created, who was a part of it, and if they need to return to it.

Screen 2



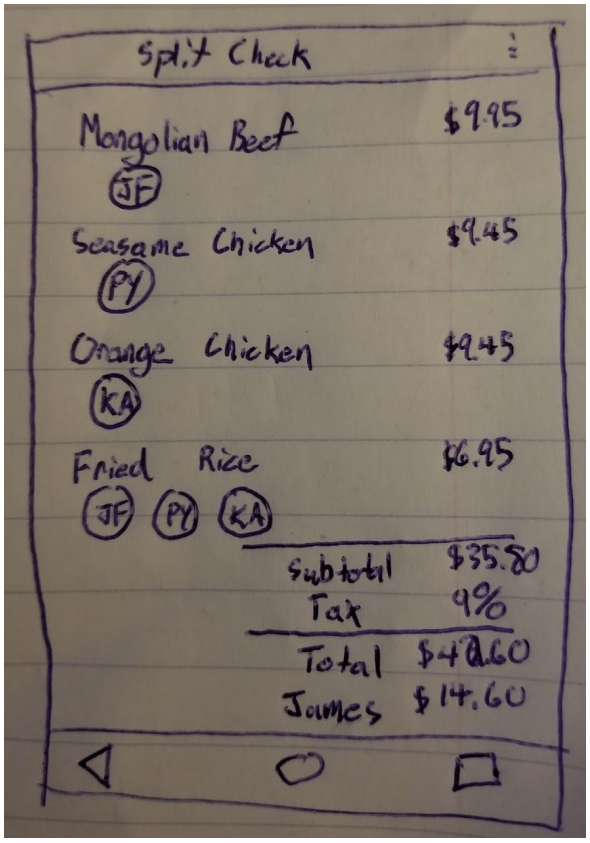
This is the activity used to add members to the group. Members will be represented in this app by circle with initials inside of it. There will also be an option to represent members with picture thumbnails.

Screen 3



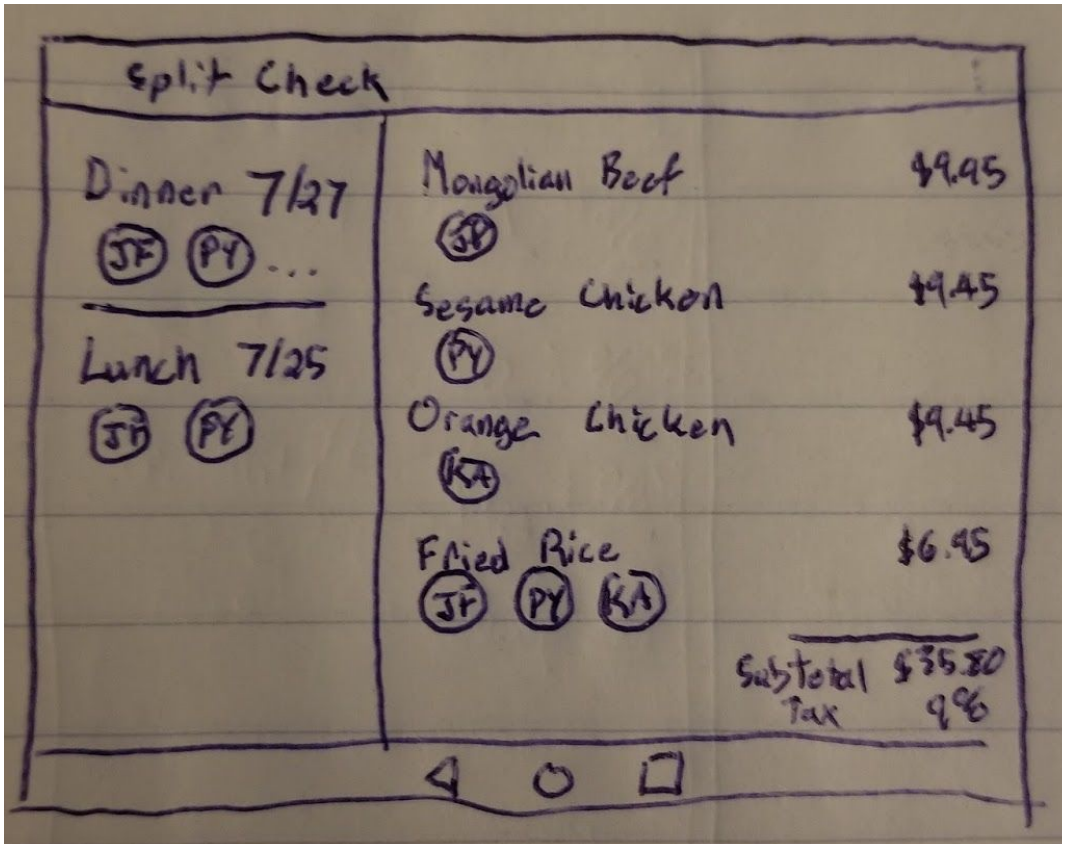
This activity will be where the user can change certain parts of the check. Things like tax will need to be adjusted for where the user is currently eating. Also, depending on where the user is eating, things like gratuity and tip may or may not be applied to the check.

Screen 4



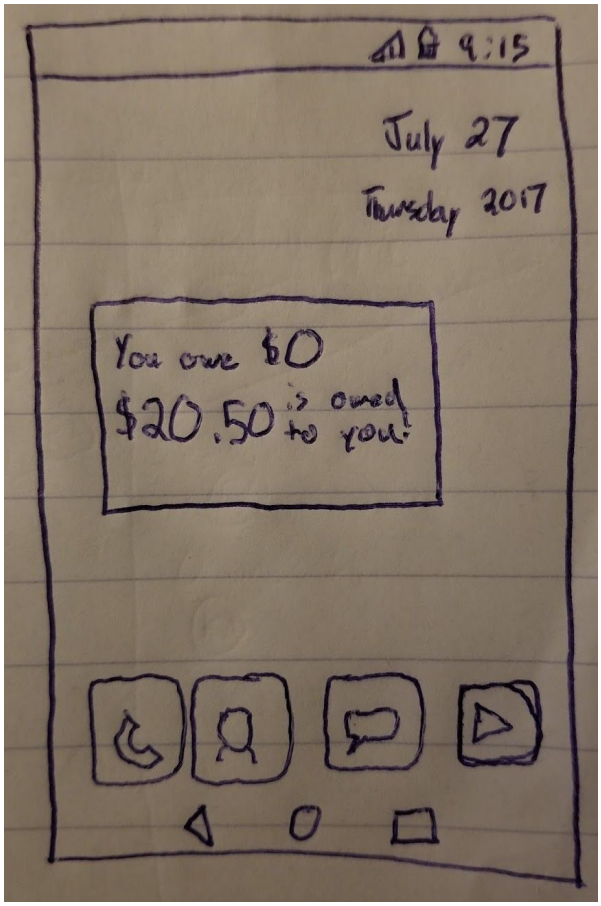
This is where the user will get to see how the check breaks down and see how much each person owes. Each added item will have a price next to it as well as who is paying for it below. At the bottom, the total will show as well as each individual's total.

Screen 5



This is how the app will look on a tablet. It will use a master/detail flow where a user can select a check and it will show fully on the right.

Screen 6



This is the home screen widget which will tell the user how much money they owe and how much is owed to them. When clicking on the widget, it will show them a breakdown what is owed to who.

Key Considerations

How will your app handle data persistence?

This app will be using local sql databases to handle data. In order to access and interact with the data, I will implement content providers.

Describe any edge or corner cases in the UX.

Edge case: User exits the app while adding items to the bill.
How it will be handled: If the user exits the app, all the previously entered info will be stored in an instance state and will be ready to use when the user come back to the app.

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

I will be using Picasso to display all images used. I will also be using the design library to implement a material design for this application.

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

I will be using Google Play Services for ads and for log in. There will be a free version with a test ad implemented and a paid version which is loaded without any ads. The google login will use their Google Account to integrate their name and profile picture into the app without having to set it up themselves.

Next Steps: Required Tasks

Task 1: Project Setup

Create java library to handle splitting of checks

- All math functions done in background quickly and efficiently

Create android libraries that handle settings for the user

- Set up a settings activity that will let the user change certain aspects of how the check is totaled
- Set up an activity which lets the user add members of a group

Set up Check class

- Check class will contain all info that is typically included on a check
- Class contains Items (e.g dishes, appetizers, drinks)
- Class contains Participants (e.g. James, Peter)

Set up Items class
Set up Participants class

Set up Firebase

- Notifies user after a period of time when money is owed to them or when they owe money to someone else

Set up Loaders

- Populate the MainActivity with local data using Loaders

Set up AsyncTask for all functions needed to be done in background

- Add search function to see be able to search for past checks

Task 2: Implement UI for Each Activity and Fragment

Create and implement the UI for the main screen

- Build UI for CheckListActivity(MainActivity)
- Implement CheckListAdapter (Shows current and previous checks)

Create and implement the UI for check setup screen

- Build UI for CheckSetupActivity
- Implement CheckSetupFragment (Handles adding items)

Task 3: Google Play Services

Implement GooglePlayServices for Ads

- Set up the correct dependencies for GooglePlayServices - Ads
- Set up Ads view

Implement GooglePlayServices for Login

- Set up the correct dependencies for GooglePlayServices - Auth

Task 4: Build Flavors

Set up free build flavor

- Implements ads

Set up paid build flavor

- Remove and dependencies and views which contain Ads

Task 5: Polish UI

Implement material design
Make sure UI is fluid and clean

- Check that animation is working as intended
- Check that styles are organized and consistent
- Check that toolbars, fabs, buttons, images are in correct areas with correct attributes

Task 6: Create Home Screen Widget

Implement a home screen widget

- Updates efficiently
- Shows money which is currently owed
- Opens app when pressed

Task 7: Build App

Get App ready for submission

- Clean
- Build and installRelease
- Sign
- Include keystore and password