

How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
 2. Name your document file: “**Capstone_Stage1**”
 3. Replace the text **in green**
-

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[Screen 1](#)

[Screen 2](#)

[Screen 3](#)

[Screen 4](#)

[Key Considerations](#)

[How will your app handle data persistence?](#)

[Describe any edge or 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 or other external services.](#)

[Next Steps: Required Tasks](#)

[Task 1: Project Setup](#)

[Task 2: Create Content Provider](#)

[Task 3: Implement UI for Screen 1](#)

[Task 4: Implement UI for Screen 2](#)

[Task 5: Implement UI for Screen 3](#)

[Task 6: Implement UI for Screen 4](#)

[Task 7: Test the application](#)

GitHub Username: bendaf

Spip - SPlit trIP expenses

Description

I would like to create a similar app to app Tricount

(<https://play.google.com/store/apps/details?id=com.tribab.tricount.android>).

The purpose of the app is to split travel bills between the travelers in a way that everyone participate only at their own expenses. The user can create trips with participants and share their expenses between them easily.

Intended User

Group members who pay bills for the others and others for him and would like to pay off debts easily.

Features

The main features of Spip.

- Manages Groups and Group members
- Stores and lists expenses
- Displays the personal balances and average spendings

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 Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

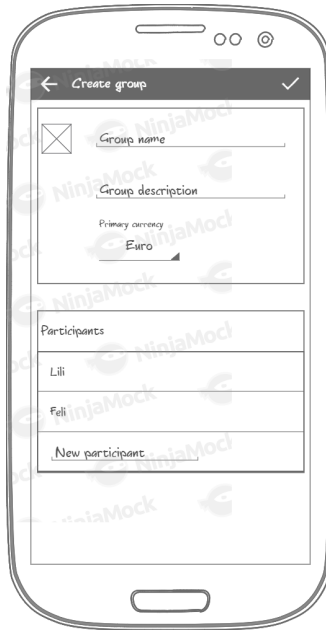
I have created my mocks in ninjamock.com: <https://www.ninjamock.com/s/CM78LTx>

Screen 1



The main screen listing the groups. Clicking on a group takes us to Screen 3 while the FAB button takes us to Screen 2

Screen 2



Create group screen. Every detail can be filled up. If a new participant name is typed than a bonus row is added to the participants recyclerview. Clicking on the back button it takes us to Screen 1 while the pipe takes us to Screen 2

Screen 3



The two tabs of screen 3: One where the balances are listed and the other where the history of the expenses. Clicking to the back button takes us to Screen 1 while the edit icon takes us to Screen 2. Clicking on the FAB takes us to Screen 4.

Screen 4



Add new expense tab. Every detail can be set then clicking on the pipe or the back arrow icon takes us to Screen 3 the difference is that the pipe adds the new expense (or edits it) while the back arrow doesn't.

Key Considerations

How will your app handle data persistence?

I will build a content provider for my database which will store the groups and expenses and members in different tables.

Describe any edge or corner cases in the UX.

The app will not support tablets, since the mobiles are more handy for this purpose. The rotation will also be disabled, most of the screens require keyboard, and they would not be usable in landscape mode.

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

I will use the android support libraries and schematic, to generate content provider code for me.

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

I will not use Google Play Services

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

- Configure libraries
- Configure AndroidManifest

Task 2: Create Content Provider

Create the database and content provider for the expense management.

- Create tests
- Create SQLite database
- Create Content Provider with the help of Schematic
- Test the Content Provider

Task 3: Implement UI for Screen 1

- Create layout for Screen 1
- Implement screen logic

Task 4: Implement UI for Screen 2

- Create layout for Screen 2
- Implement screen background logic

Task 5: Implement UI for Screen 3

- Create layout for Screen 3
- Implement screen logic

Task 6: Implement UI for Screen 4

- Create layout for Screen 4
- Implement screen logic

Task 7: Test the application

- Test the ui functionality
- Test stability

Add as many tasks as you need to complete your app.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
 - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"

- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"