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
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: Your Next Task
  Task 4: Your Next Task
  Task 5: Your Next Task

GitHub Username: joycelin12

Dream Destinations

Description

Building a travel app that is able to search all the flights from all the different countries. It would be able to save the trip where a notification will be sent whenever there is a price drop. This will help user to keep track of good price of the trip. When there is a good sale, able to share to friends.

Intended User

This app is for travelers who are too busy to keep track of the price of their trip everyday to get the best deal.

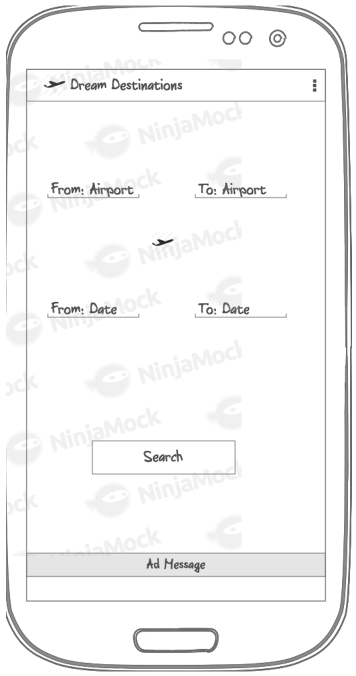
Features

Search for flights around the world.
Save the flights to alert user if there is a price drop

User Interface Mocks

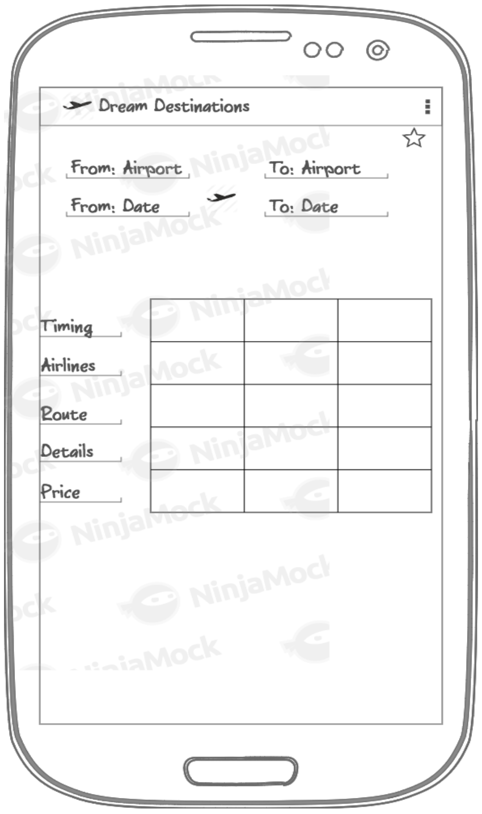
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1



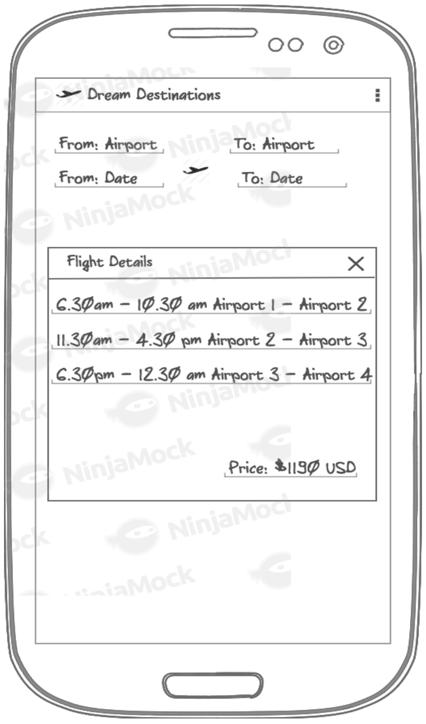
Main screen for searching the flights. For the airport fields, it will be a drop down to select the airport name. For the dates, it will be date picker to select the dates.

Screen 2



After the search, the display of the flight information will be displayed in table form as above. Can click on the star to save the trip and get price alert on the trip if there is a price drop. If there are more flights it is scrolled to the right.

Screen 3



6.30am – 10.30 am Airport 1 – Airport 2

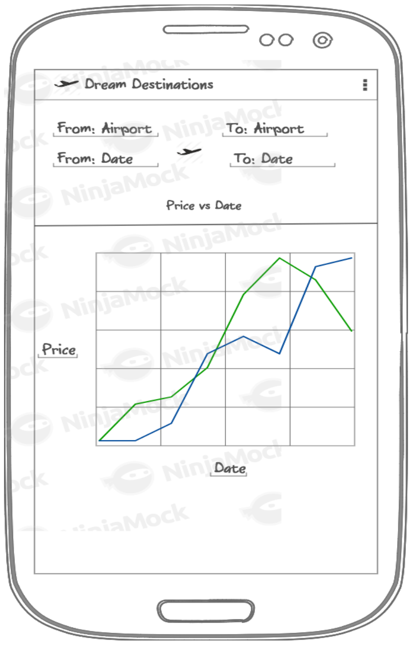
11.30am – 4.30 pm Airport 2 – Airport 3

6.30pm – 12.30 am Airport 3 – Airport 4

Price: $1190 USD

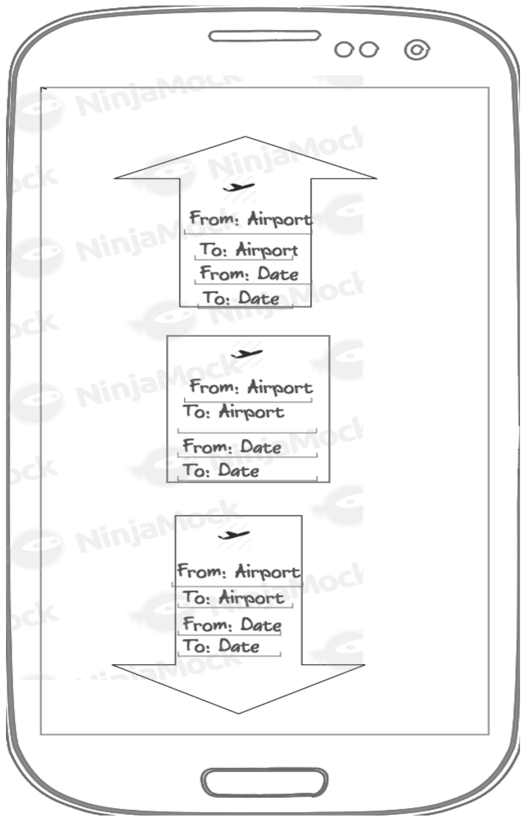
Can click on the details to screen 2 and display the dialog box in screen 3 to display the details of the flight selected.

Screen 4



This will show the price changes over the days for the saved trip from screen 2.

Screen 5



These 3 symbols are examples of the widgets. The first one will be showing the price of the saved trip is increasing, the middle one is showing there is no difference for the saved trip and the last one is showing the price of the saved trip is decreasing.

Key Considerations

How will your app handle data persistence?

In order to save the trip to keep track for price alert, I will be building a content provider with sqlite database.

Describe any edge or corner cases in the UX.

If there is no internet connection, once the search button is pressed, there will be a toast message that says there is no internet connection.

If it is offline, it would not be possible to search for possible flights. However, it can display the saved trip graphs to show the trend of the price for the saved trip.

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

Picasso will be used to handle the loading and caching of images.

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

I will be using location to get the current location for the airport. I will also be using admob to display advertisements in all the pages of the app.

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

Java programming language will be used in the creation of the app.
The following versions of android studio and gradle are used.
Android studio 3.0.1
Gradle 3.0.1

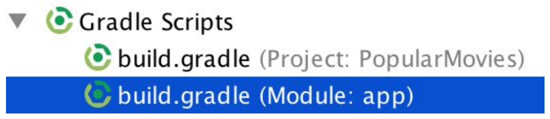
Referencing from:
https://docs.google.com/document/d/1ZIN1fUsCSKuInLECcJksIIqvpKIP7jWL2TP9m6UiA6I/pub?embedded=true

a. Image Library - Picasso

How to Setup Picasso

We recommend that this project use Picasso, a powerful library that will handle image loading and caching on your behalf. If you prefer, you're welcome to use an alternate library such as Glide.
We've included this to reduce unnecessary extra work and help you focus on applying your app development skills.

You'll need to modify the build.gradle file for your app. These modifications will happen in the build.gradle file for your module's directory, *not* the project root directory (it is the file highlighted in blue in the screenshot below).


Gradle Scripts
build.gradle (Project: PopularMovies)
build.gradle (Module: app)

In your app/build.gradle file, add:

```
repositories {
    mavenCentral()
}
```

Next, add **compile 'com.squareup.picasso:picasso:2.5.2'** to your dependencies block.

Using Picasso To Fetch Images and Load Them Into Views

You can use Picasso to easily load album art thumbnails into your views using:
`Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imageView);`
Picasso will handle loading the images on a background thread, image decompression and caching the images.

b. Add dependencies for google play services for admob and location.

c. Working with the Skyscanner API using rapidapi

Skyscanner Flight Search.Create session

Create a flight search session. A successful response contains no content. The URL to poll the results is provided in the Location header of the response.

Skyscanner Flight Search.Poll session results

Get itineraries from a created session

d. Adding graphview for the graphs of the price alert from the following website. http://www.android-graphview.org/ using version graphview:4.2.2

e. App includes support for accessibility. That includes content descriptions and make touch targets large.

f. App keeps all strings in a strings.xml file and enables RTL layout switching on all layouts.

Task 2: Implement UI for Each Activity and Fragment

Build UI for MainActivity and the fragment for the first page.

- Create layout according to the mock ups.

Build UI for ResultActivity and the fragment for the second page.

- Create layout according to the mock ups.
- Create the dialogs for the third page.

Build UI for TrackerActivity for the fourth page.

- Create layout according to the mock ups.

App uses standard and simple transitions between activities.

Task 3: Implement Google Play Services

Add the dependencies for the google play services and try to implement the location and admob in the main activity.

Task 4: Create the asynctask for the flight search query.

Build the asynctask class to get the flight search query using the skyscanner api in rapidapi. This will get the results of the search and fill up the ResultActivity with the results of the search.

Task 5: Add price alert to the trip

If the star is being clicked, it will save the trip to the database where content provider is used. It will be scheduled every day to retrieve the new prices everyday. If the price drop, it will alert the user about the price drop as well as display in the widget the price drop.

Task 6: Building the graphview for TrackerActivity

The app uses a Loader to move its data to its views by building the graph to showcase the price alert over the dates.

Task 7: Build the share button when there is a good sale

The app will share trips with good price to friends.

Task 8: Create the widget to display the price alert of the app

Use the wizard to create the necessary files for the widget creation with 3 different symbols with the different price trend.

Task 9: Create the signing configuration

App is equipped with a signing configuration, and the keystore and passwords are included in the repository. Keystore is referred to by a relative path.

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**"