GitHub Username: ramji93

Travel Easy

Description

Travel Easy is a app for flight booking that searches and populates list of different itineraries available for a given flight journey ordered according to traveller's preferences (like price, duration). On drilling down a chosen itinerary, the app populates list of numerous travel agencies covering the same journey along with their price. Users can go to the booking link of any of the listed agencies and book tickets or can share its booking link via sharing apps. (NOTE: Only for outbound flights)

Intended User

Travellers who are tired of visiting each and every travel agency's site to know about the price for the journey.

Features

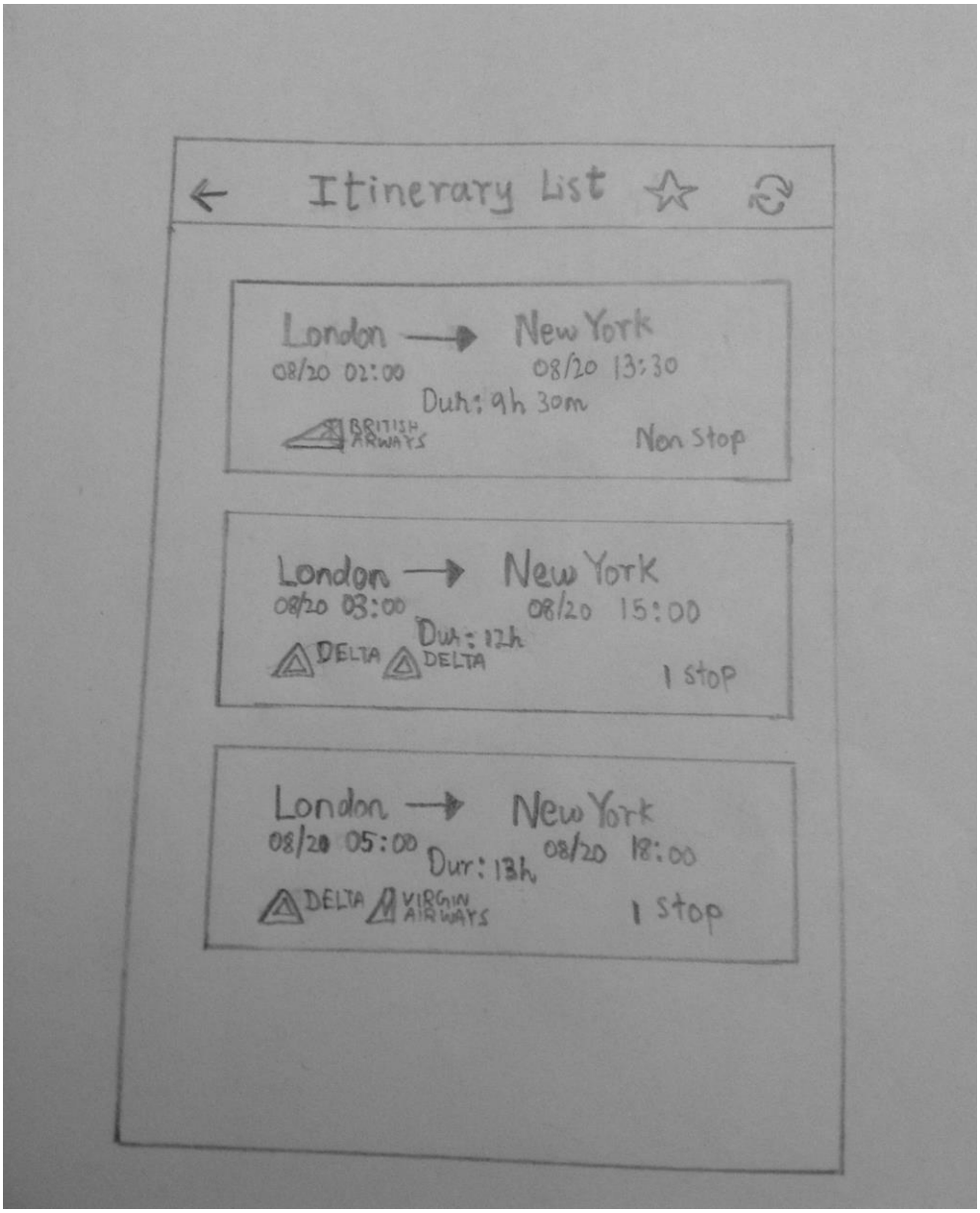
- Lists different itineraries just by entering the origin, destination and date of journey.
- Options to sort the itineraries list by user preferences (price, duration, arrival time).
- Displays all the travel agencies supporting the selected itinerary along with their prices at one place.
- Option to save a flight search and use it in the future.

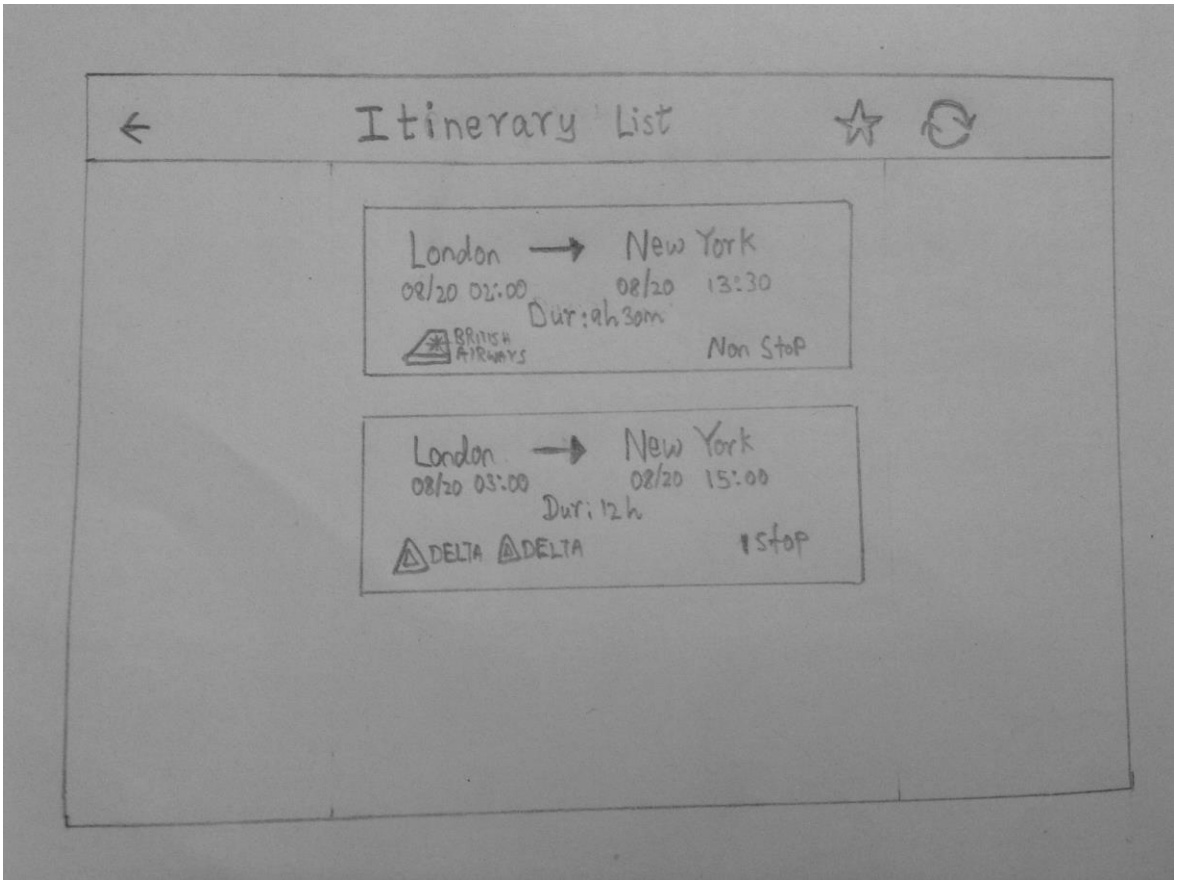
User Interface Mocks

Screen 1

This is the launcher activity for the app. It can also be reached by clicking flight search in navigation drawer. It contains two autocomplete fields for getting origin and destination from the user, date picker tool for journey date, spinner for other details like class, number of adults and children. The autocomplete field gets suggestions from google location services as the user type. The navigation drawer will be shown below.

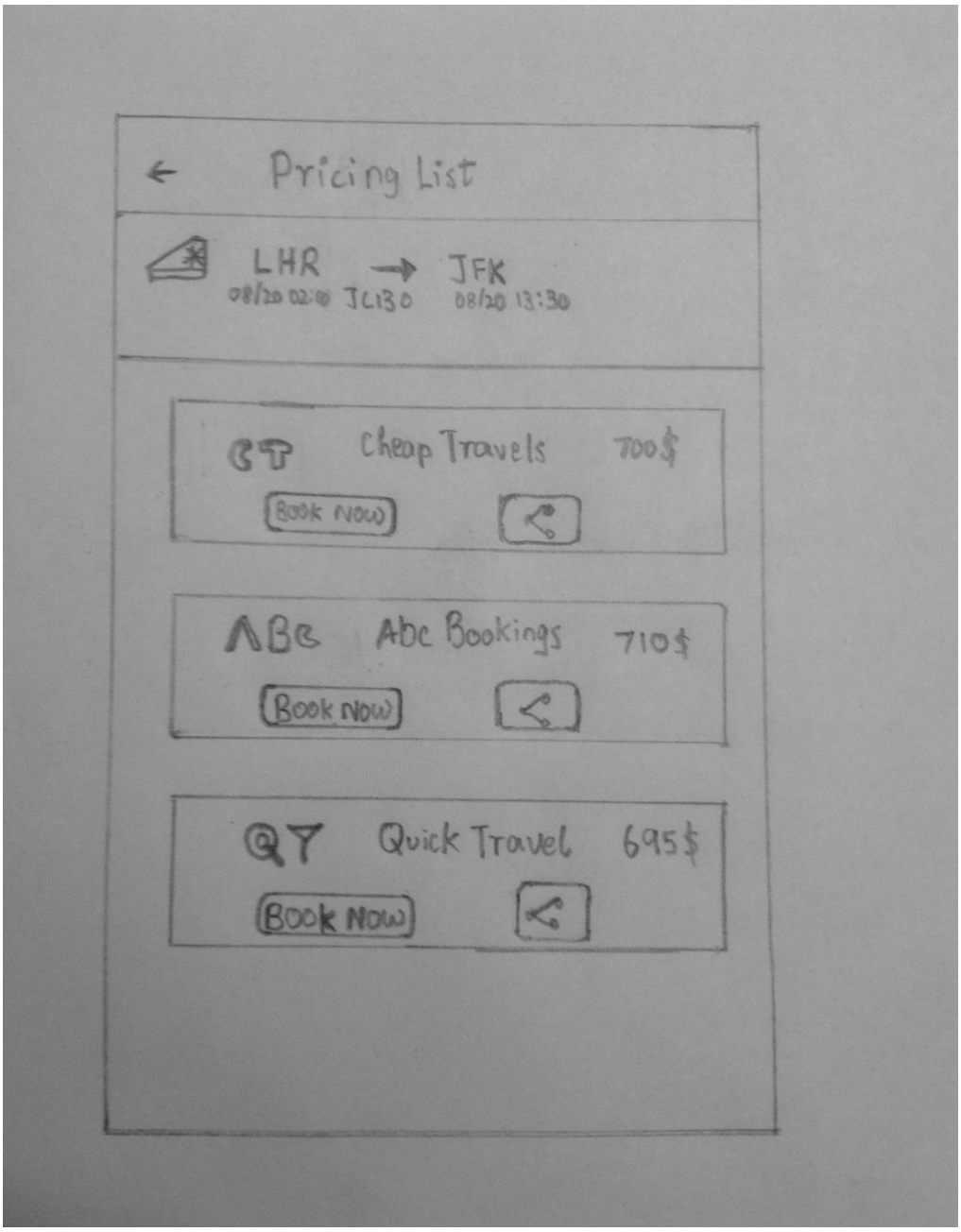
Screen 2

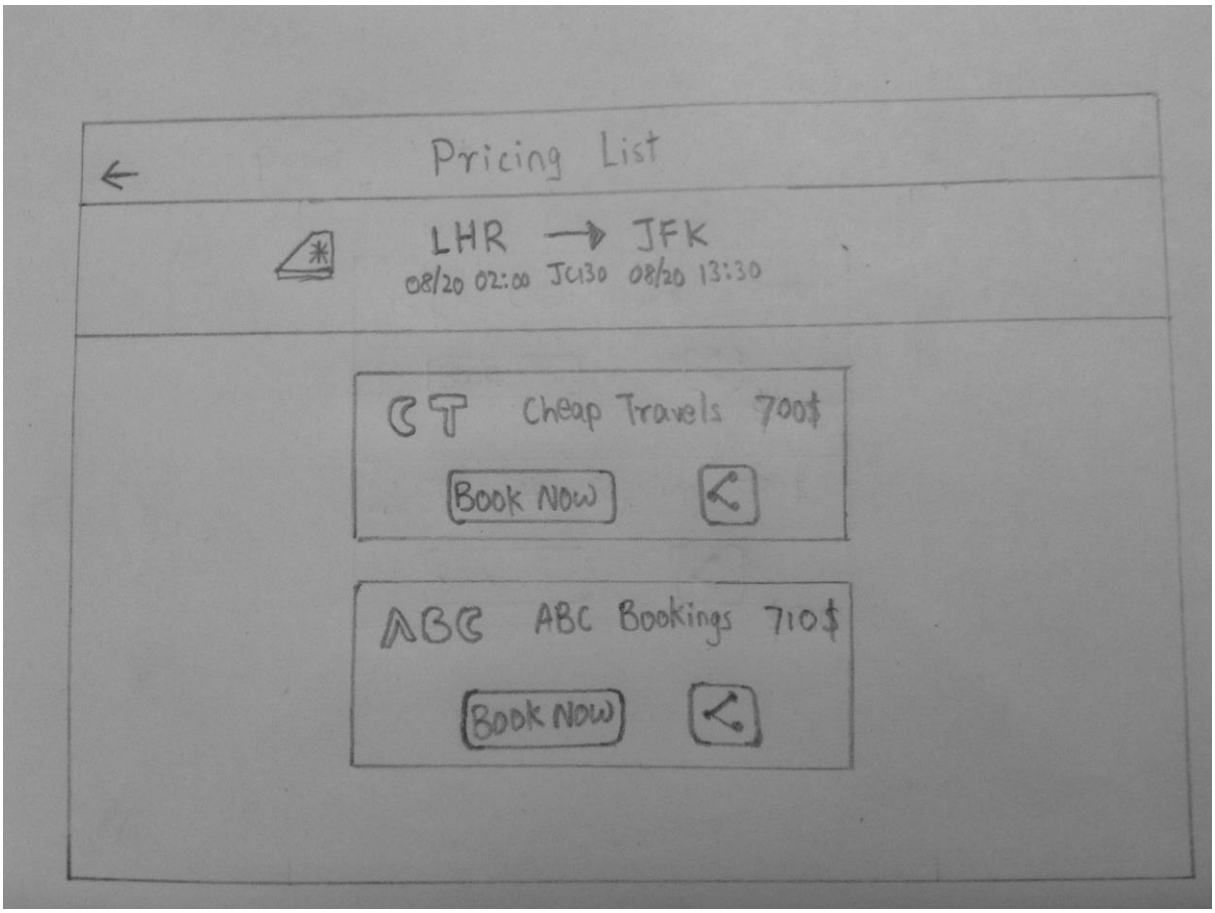

Itinerary List
London → New York
08/20 02:00
08/20 13:30
Dur: 9h 30m
BRITISH AIRWAYS
Non Stop
London → New York
08/20 03:00
08/20 15:00
Dur: 12h
DELTA
DELTA
1 stop
London → New York
08/20 05:00
Dur: 13h
08/20 18:00
DELTA
VIRGIN AIRWAYS
1 stop



This screen will be shown as a result of clicking search button in previous screen. It populates different itineraries for given search parameters. Each item contains information such as departure, arrival, number of stops, duration and also carrier logos. The app bar contains save option (star icon) to save the search parameters and use it for the future without typing it again. It also contains sync option to sync the contents immediately to get the updated list itineraries.

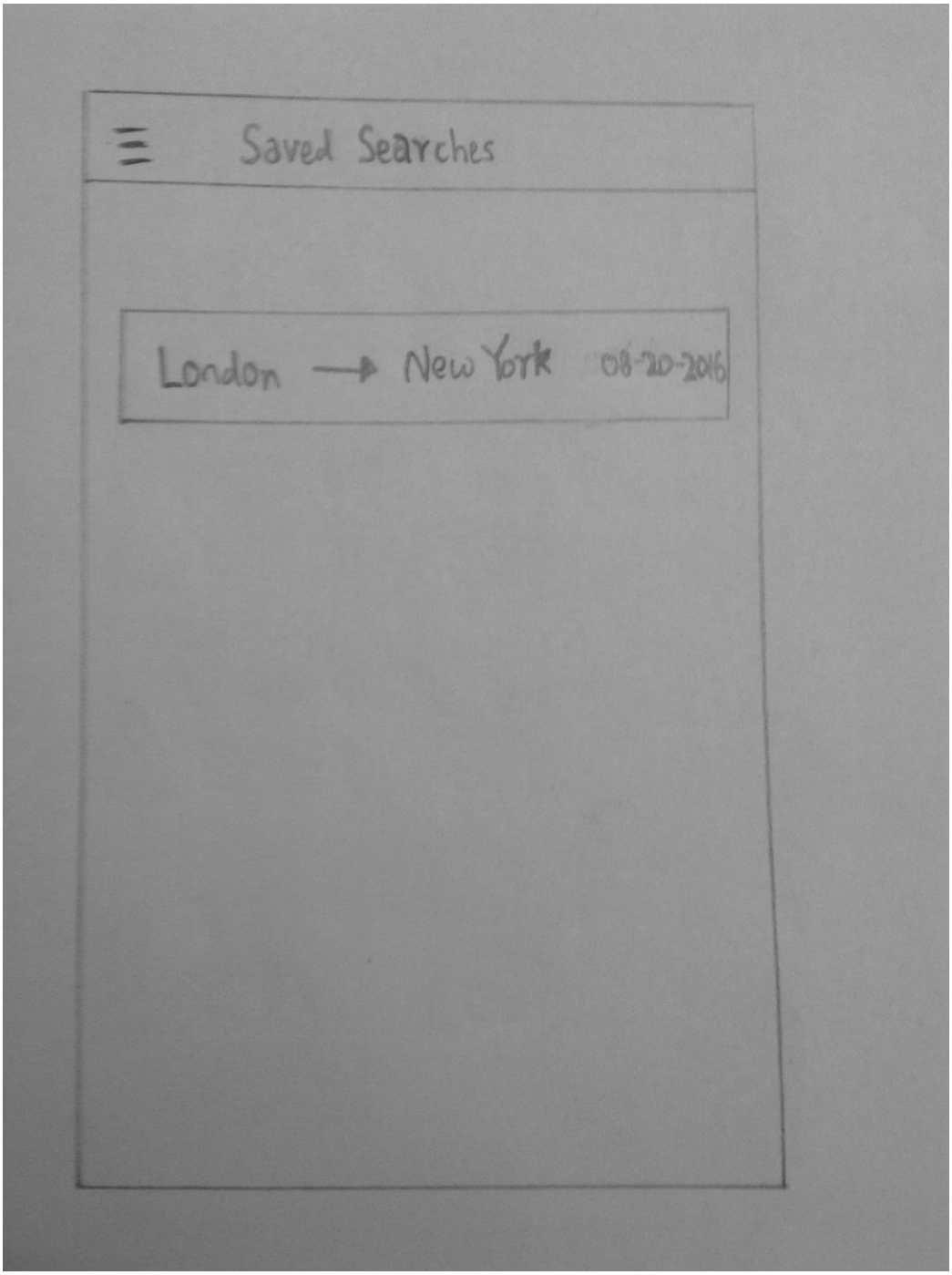
Screen 3

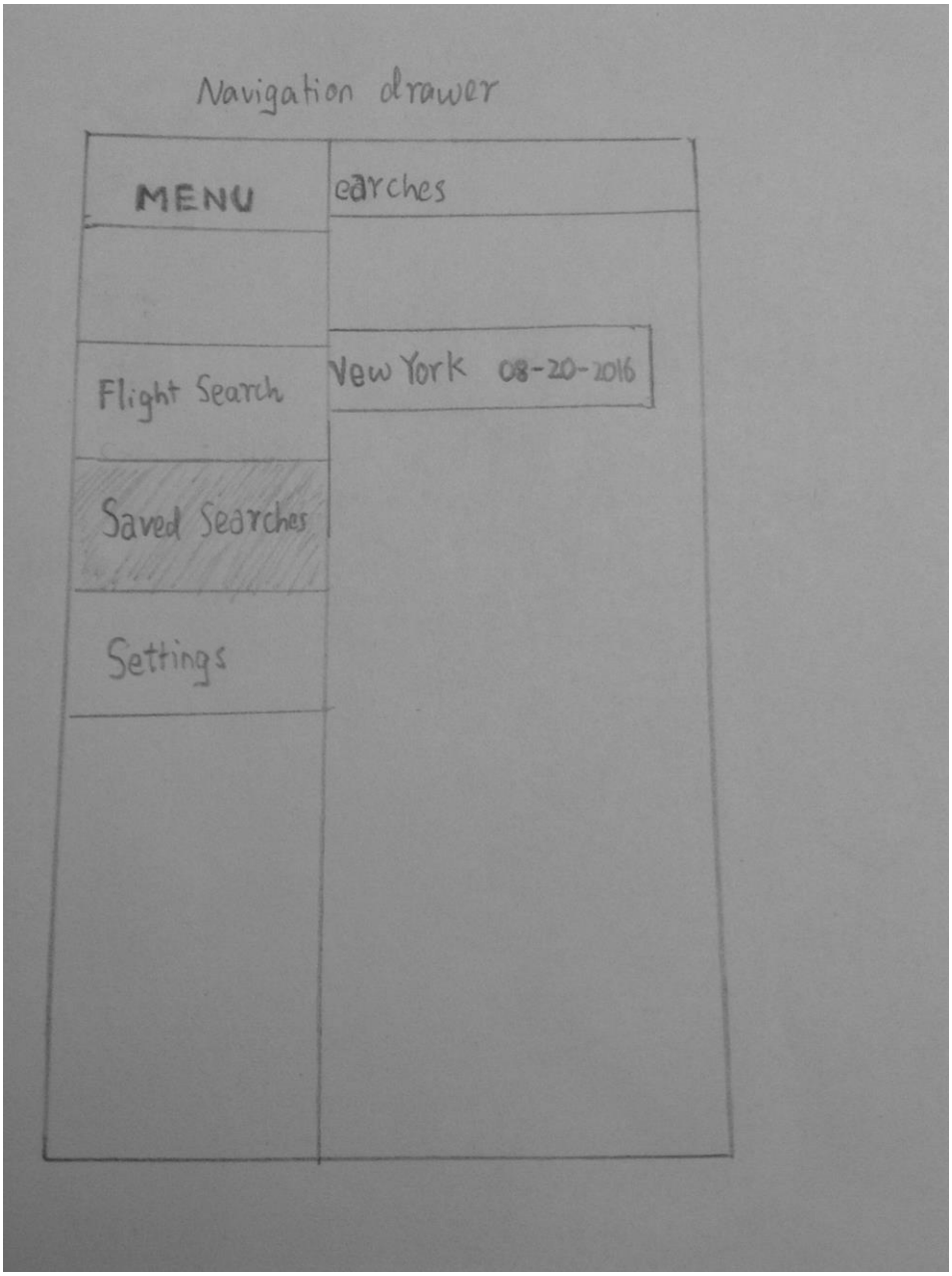

Pricing List
LHR
JFK
JC130
Cheap Travels
700$
Book Now
Abc Bookings
710$
Book Now
Quick Travel
695$
Book Now



This screen will be displayed on clicking any list item (i.e. itinerary) in the previous screen. It populates each flight travel involved in the selected itinerary at the top and list of all the travel agencies covering this itinerary along with its price. Each item in the list has book now option which takes the user to the booking site of the agent and share option which allows the user to share the booking link via sharing apps.

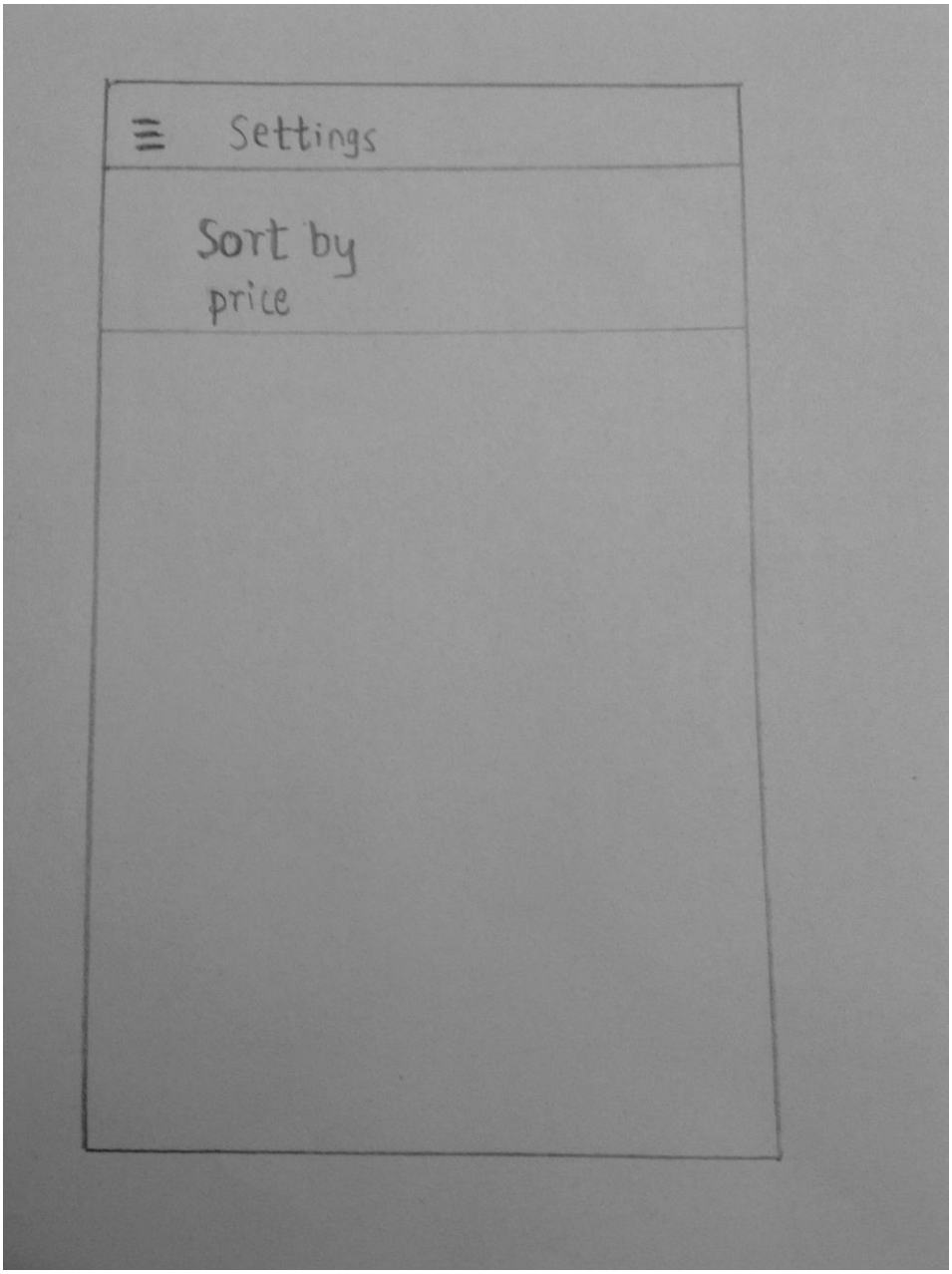
Screen 4





This screen will be displayed on clicking ‘saved searches’ item in navigation drawer. It displays list of saved searches. On clicking a list item, the itinerary list for the saved search parameters will be displayed. Note that itinerary list may not be the same as in the time of saving and depends on the situations.

Screen 5



This screen can be reached by selecting settings in navigation drawer. It allows the user to save their preference over the sorting option. It contains a list preference that makes itinerary list to be sorted by either price, duration or arrival time.

Key Considerations

How will your app handle data persistence?

- For populating itineraries list and travel agencies list, we use Sync adapter as we need it real time and there is no point in using content provider for this.

- For storing search parameters and populating the saved searches, we use content provider (using SQLlite db).
- Shared Preference is enough to store sorting option.

Describe any corner cases in the UX.

This app contains a widget for getting to the itineraries list screen inside the app directly from phone/tablet home screen by entering search parameters like origin, destination, date etc. on the widget.

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

Glide to handle the loading and caching of carrier logos.

Describe how you will implement Google Play Services.

- Google location services to provide suggestions when the user is typing in the origin or destination autocomplete text fields.
- Google Admob services to display ads between itineraries screen and agencies list screen.

Next Steps: Required Tasks

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

Task 1: Project Setup

- Create the project with appropriate initial configurations such as min sdk, target sdk.
- Include the relevant libraries (glide) and support libraries in app gradle file.
- Get the unique api key from the api provider (skyscanner).

Task 2: Implement UI for Each Activity and Fragment

- Create layouts for each activity and fragment.
- Implement the navigation drawer along with it.
- Make sure theme of the app is appcompat or new theme derived from appcompat.

Task 3: Implement Google Location Services and Admob

- Include google play services location and admob library in app gradle file.
- Get google api key and mention it in app manifest for location sevices.
- Implement their api in the code.

Task 4: Implement Asynctask to retrieve data and process them

- Implement asynctask to retrieve relevant data.
- Create utility methods to process them.
- Display important information in recycler view as in the mock screens.

Task 5: Create new Content provider and implement data Loader

- Create new content provider to store and retrieve the saved searches.
- Implement data loader (cursor loader) to display the saved searches.

Task 6: Handle errors

- Handle the errors in the user input and in the server's response to prevent the app from crashing.

Task 7: Create Widget for the app

Task 8: Support for Accessibility