

## Description

## Intended User

## Features

## User Interface Mocks

Screen 1: Splash Screen

Screen 2 : Search Flights Screen

Screen 3 : Flights (Search Results)

Screen 4 : Flight Details

Screen 5 : Flight Details with Options Menu Open

Screen 6 : Saved Flights

Screen 7 : Navigation Drawer

Screen 8 : Airplanes

Screen 9 : Widget

## 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.

## Required Tasks

Task 1: Prepare UI Mockups and Project Plan

Task 2: Project Setup

Task 3: Test Rest Client (Retrofit)

Task 4: Generate POJOs with JSON Request and Responses

Task 5: Implement UI for Each Activity and Fragment

Task 6: Update Retrofit calls to Get Available flights

Task 7: Create Custom Adapters

Task 8: Setup Google Maps

Task 9: Setup SQLite and Content Provider

Task 10: Add a Widget

**GitHub Username:** [akndmr](#)

# Tourlines App

## Description

Tourlines, consumes Turkish Airlines API to show Available Flights to the users. Users can search flights after choosing Departure Time and Airport, Arrival Time and Airport, number and type of passengers. Results matching with the search queries will be shown. Tourlines also provides detailed information about the plane like Plane Model and other technical details of the airplane. User can also see Departure City and Destination City on the map.

## Intended User

This app is for users who often search for flights to find the cheapest or available ones. Also users who are curious about technical details of airplanes of Turkish Airlines.

## Features

Main features of your app.

- Lists available flights of Turkish Airlines
- Shows Departure and Destination cities on the map
- Shows original images of airplanes
- Shows technical details of the airplane
- App is coded in Java language.
- App keeps all String resources in string.xml
- App supports right-to-left languages (enabled RTL for all layouts)
- App has content descriptions for all images used (Accessibility)
- App supports tab navigation (Up, Down, Right, Left)
- App uses IntentService for webservice calls
- App uses Cursor Loaders to access database

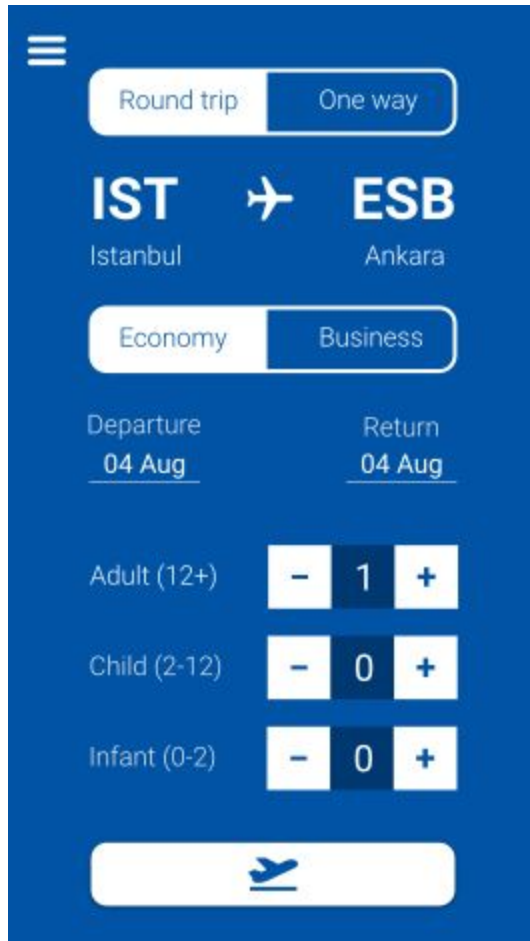
## User Interface Mocks

### Screen 1: Splash Screen



Splash Screen

## Screen 2 : Search Flights Screen



The screenshot shows a flight search interface on a blue background. At the top left is a hamburger menu icon. Below it are two buttons: "Round trip" (selected) and "One way". The origin is "IST" (Istanbul) and the destination is "ESB" (Ankara), separated by an airplane icon. Below these are two buttons: "Economy" (selected) and "Business". The departure date is "04 Aug" and the return date is "04 Aug". There are three rows for passenger counts: "Adult (12+)" with a count of 1, "Child (2-12)" with a count of 0, and "Infant (0-2)" with a count of 0. Each row has minus, plus, and reset buttons. At the bottom is a large white button with a blue airplane icon.

Menu

Round trip One way

IST ✈ ESB  
Istanbul Ankara

Economy Business

Departure Return  
04 Aug 04 Aug

Adult (12+) - 1 +

Child (2-12) - 0 +

Infant (0-2) - 0 +

Search

User enters required informations to search matching flights.

## Screen 3 : Flights (Search Results)



User can see listed results of search for flights. When user taps on an result item, user navigates to Flight Details screen.

## Screen 4 : Flight Details



User can see a map where Departure and Destination cities pinned and a line drawn between cities. Below the map, there is technical information about the airplane of this flight and below that user can find same information in the Flights screen which includes, Departure and Arrival times, Flight Duration and Type, Passenger details and Price.

## Screen 5 : Flight Details with Options Menu Open



User can either Save or Share flight details using the Options Menu on top left corner of the screen.

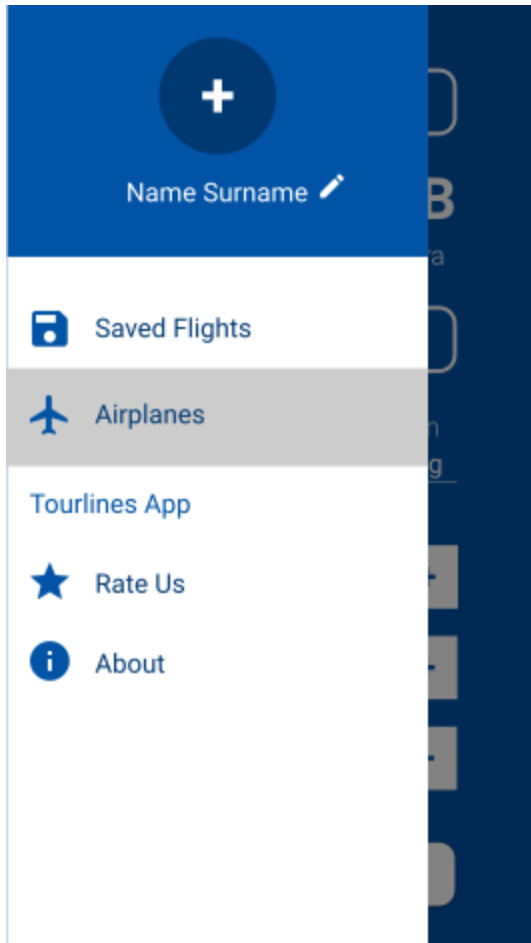
## Screen 6 : Saved Flights



User can see Saved Flight details here.



## Screen 7 : Navigation Drawer



User can navigate to Saved Flights and Airplanes and their technical details of Turkish Airlines screens. Users also may navigate to Play Store page of the app via Rate Us link and see information about app and creator in About Screen.

## Screen 8 : Airplanes

**AIRPLANES**

### A340-300



Airbus A340-300 wide-body specifications

Passenger Capacity	270-354
Maximum take-off weight	257,000 kg
Wing span	60.30 m
Body length	63.68 m
Height	16.99 m
Maximum cruising altitude	12,500 m/41,000 ft
Maximum cargo capacity	44,836 kg/152.80 m³

### B777-300

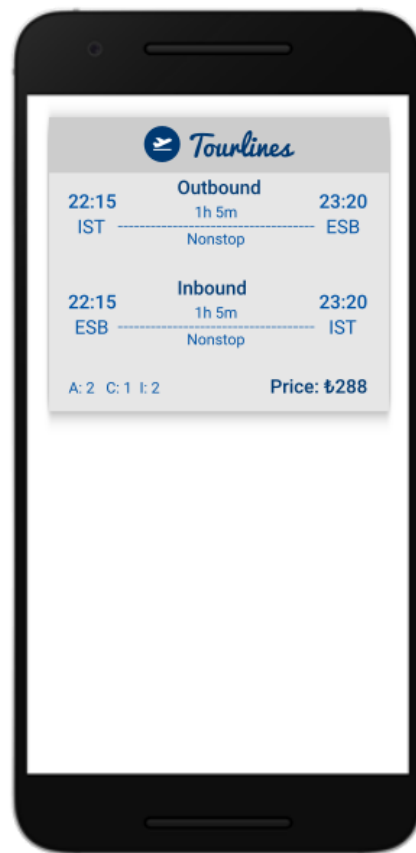


Boeing B777-300 ER specifications

Passenger Capacity	349-400
Maximum take-off weight	351,534 kg
Wing span	64.80 m
Body length	73.90 m
Height	18.60 m
Maximum cruising altitude	13,140 m/43,100 ft
Maximum cargo capacity	57,784 kg/213.85 m³

Users can see technical details of the airplanes of Turkish Airlines.

## Screen 9 : Widget



## Key Considerations

### How will your app handle data persistence?

App will use Content Provider with SQLite Database to save Flights.

### Describe any edge or corner cases in the UX.

If there is no internet connection, Users can choose to navigate to Saved Flights or Airplanes screens.

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

**Picasso (v2.5.2)** to handle the loading and caching of images.

**ButterKnife (v8.8.1)** to handle field and method binding for views.

**Retrofit (v2.4.0)** to handle to retrieve and upload JSON via REST based webservice.

**GSON (v2.4.0)** to convert JSON to POJO and vice versa.

**Lottie (v2.5.5)** for high quality animations.

**Play Services Maps (v15.0.1)** for showing cities on the map.

**Firebase Authentication (v16.0.2)** for authentication.

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

Maps service will be used to show Departure and Destination Cities on the map and draw a line between two cities.

Firebase Authentication to authenticate users to use app.

## Required Tasks

### Task 1: Prepare UI Mockups and Project Plan

- Write a plan which has requirements and goals of the project.
- Draw sketches of UI and colorful Mockups of the screens.

### Task 2: Project Setup

- Add dependencies of the libraries.
- Fix any conflict between libraries.

### Task 3: Test Rest Client (Retrofit)

- Test retrofit if it's successfully making API calls with dummy json.

### Task 4: Generate POJOs with JSON Request and Responses

Use [www.jsonschema2pojo.org](http://www.jsonschema2pojo.org) to auto-generate POJOs for JSON Request and Response.

## **Task 5: Implement UI for Each Activity and Fragment**

- Build UI for MainActivity
- Build UI for Flights
- Build UI for Flight Details
- Build UI for Saved Flights (copy of Flights)
- Build UI for Airplanes
- Build UI for About

## **Task 6: Update Retrofit calls to Get Available flights**

- Check if Retrofit calls return successful response and bind(Gson) it to POJOs

## **Task 7: Create Custom Adapters**

- Create a custom adapter to show flights
- Create a custom adapter to show flight details
- Create a custom adapter to show Airplanes

## **Task 8: Setup Google Maps**

- Add Google Play Services
- Get a Google Maps API key
- Create Layout for map

## **Task 9: Setup SQLite and Content Provider**

- Create DB Contract and Content Provider class
- Expose application data to widgets

## **Task 10: Add a Widget**

- Show latest saved flight info in widget