GitHub Username: appdevgenie

Shuttle Service

Description

Need a shuttle to the airport or visit someone in the next town? Book a seat with Shuttle Service. Daily trips between Mpumalanga and Johannesburg Airport, with 9 pick-up/drop-off points.

Intended User

Application is aimed at commuters.

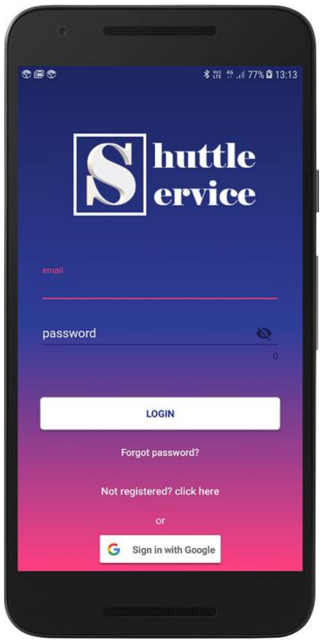
Features

Main features:

- App is written solely in the Java Programming language
- Available to all users whether registered or not:
    - Route and stops, including time to next stop
    - Price list
    - Contact info
- Check available seats
- Make booking
- View booking history
- Weather forecast

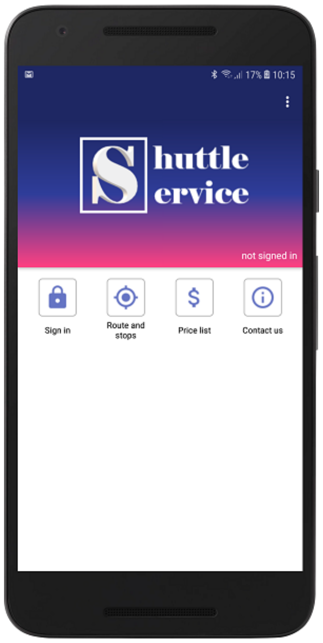
User Interface Mocks

Screen 1



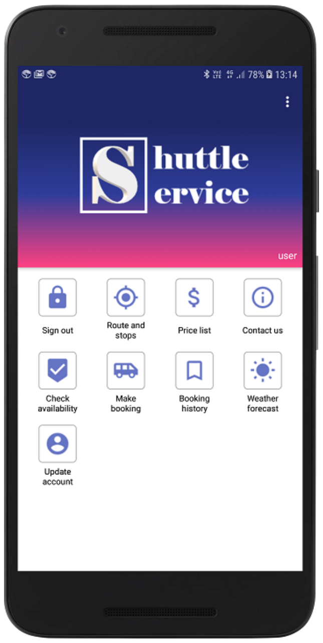
Login screen: Firebase authentication (email or Google)

Screen 2



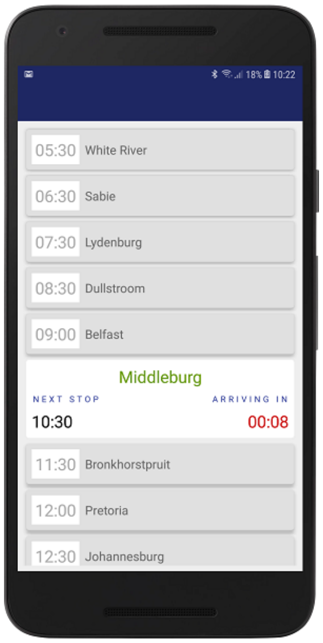
Logged out default screen: users can view daily route and price check

Screen 3



Logged in user: users can view daily route, price check, availability, make booking, view bookings, update account information and get latest weather forecast

Screen 4



05:30 White River

06:30 Sabie

07:30 Lydenburg

08:30 Dullstroom

09:00 Belfast

Middleburg

NEXT STOP 10:30

ARRIVING IN 00:08

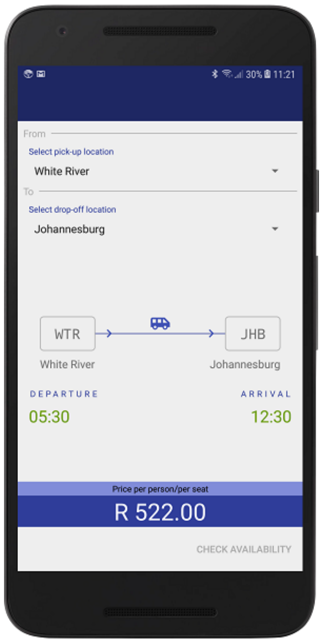
11:30 Bronkhorstpruit

12:00 Pretoria

12:30 Johannesburg

Route and stops screen: users can view daily route and next stop

Screen 5



From

Select pick-up location: White River

To

Select drop-off location: Johannesburg

DEPARTURE 05:30

ARRIVAL 12:30

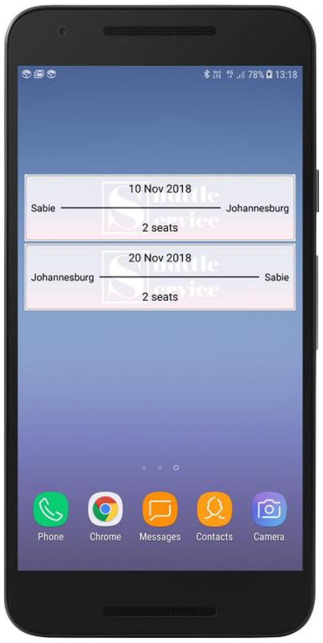
Price per person/per seat

R 522.00

CHECK AVAILABILITY

Price check screen: users can view price between any two towns

Screen 6



Home screen widgets

Key Considerations

How will your app handle data persistence?

Booking, travel and user data will be saved on Firebase Firestore. Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. It keeps your data in sync across client apps through realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. All data caching is handled by Firebase. All tasks are asynchronous and will not affect other code.

The application also has a weather forecast activity included. The weather data is collected online as JSON and populated in a recyclerview. These short on-demand JSON requests are done in an AsyncTask

Describe any edge or corner cases in the UX.

User will be able to move between various fragments using buttons located at the bottom of views and to main menu using the back button.
Toast messages will be used when new bookings are loaded successfully, booking info is added to widget and not connected to an active internet connection

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

- com.android.support:appcompat-v7:27.1.1
- com.android.support.constraint:constraint-layout:1.1.3
- com.android.support:support-vector-drawable:27.1.1
- com.android.support:support-v4:27.1.1
- com.android.support:recyclerview-v7:27.1.1
- com.android.support:design:27.1.1
- com.google.android.gms:play-services-auth:16.0.1
- com.squareup.picasso:picasso:2.5.2
- com.android.support:cardview-v7:27.1.1
- com.jakewharton:butterknife:8.8.1
- com.google.firebase:firebase-core:16.0.4
- com.google.firebase:firebase-auth:16.0.5
- com.google.firebase:firebase-database:16.0.4
- com.google.firebase:firebase-storage:16.0.4
- com.google.firebase:firebase-firestore:17.1.2
- com.google.gms.google-services
- com.android.tools.build:gradle:3.2.1
- com.google.gms:google-services:4.1.0

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

Firebase requires play services which will be added to gradle.

Next Steps: Required Tasks

Task 1: Project Setup

Project setup:
- Create new Android Studio project
- Configure libraries
- Include Firebase to application

Task 2: Implement UI for Each Activity and Fragment

List the subtasks:

- Build UI for Main Activity: use single activity and recyclerview for both signed in and signed out users with different actions
- Build UI for Login Activity: application should start in MainActivity even if user is not logged in so that route and price list can be viewed by all
- Build UI for Routes Stops fragment: create recyclerview view type for default view and next stop
- Build Ui for Price List fragment: commuter can travel between any two towns, show price and times
- Build UI for Contact Us fragment: include admin and driver info ICE
- Build Ui for Check Availability fragment: get seats available for given date
- Build UI for Make Booking fragment: record all user and travel info
- Build UI for Booking History fragment: info will be collected from firebase
- Build UI for Update User Account Info fragment: update to firebase
- Add content descriptions to all imageviews
- Add d-pad navigation to recyclerviews
- Enable RTL switching in layouts

Task 3: Firebase

List the subtasks:

- Implement Firebase authentication and firestore
- Ensure that Firestore rules are correct: only authenticated users can access data

Task 4: Main and login

List the subtasks:

- Create Java classes for Main Activity and Login Activity
- Email will be used as main identification, collect from authentication
- Ensure that application opens on recycerview grid
- Sign in option on main screen as icon
- Add option menu to sign out and close application

Task 5: Route

List the subtasks:
- Create Java classes for Route and Stops fragment
- Use recyclerview to show daily trips and stops
- Change view type of next stop view, use runnable to update in real time, remove callback when fragment is destroyed

Task 6: Weather forecast

List the subtasks:
- Register to use online weather API (i.e. OpenWeatherMap)
- Create NetworkUtils class to collect JSON data
- Use AsyncTask to populate

Task 7: Link up booking activities

List the subtasks:
- Create Java classes for Price List, Check Availability and Make Booking fragments
- Link fragments so that user can move between fragments
- Use firebase to collect user info and ensure all booking info is correct before proceeding

Task 8: Master detail flow

List the subtasks:
- Create layouts for various screen sizes and orientations

Task 9: Add widget

List the subtasks:
- Create home screen widget so that user can view specific booking history

Task 10: Clean up code

List the subtasks:
- Create common styles for reused views
- Ensure all string values are included in the strings.xml file
- Remove all unused classes and xml info