Capstone Stage 1

GitHub Username: abhinituk

All-in-one

Description

It’s always good when you have a single destination app for all your needs. All-in-one provides user a single destination for all needs. User can book cabs, order food, view nearby places etc. Storage is a real problem for most Indians. The Android system and WhatsApp consume 60-70 percent of the total space on regular android phones, which makes All-in-one an essential app to have.

Intended User

This app is intended for everyone who wants a single destination for all their needs.

Features

List the main features of your app. For example:

- User can book cabs using all-in-one app.
- User can order food from their near by restaurant using all-in-one app.
- User can search for nearby places using all-in-one app.
- User can search for nearby restaurant using all-in-one app.
- User can view latest cricket score and commentary.
- User can view latest offers available.

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 Photoshop or Balsamiq.

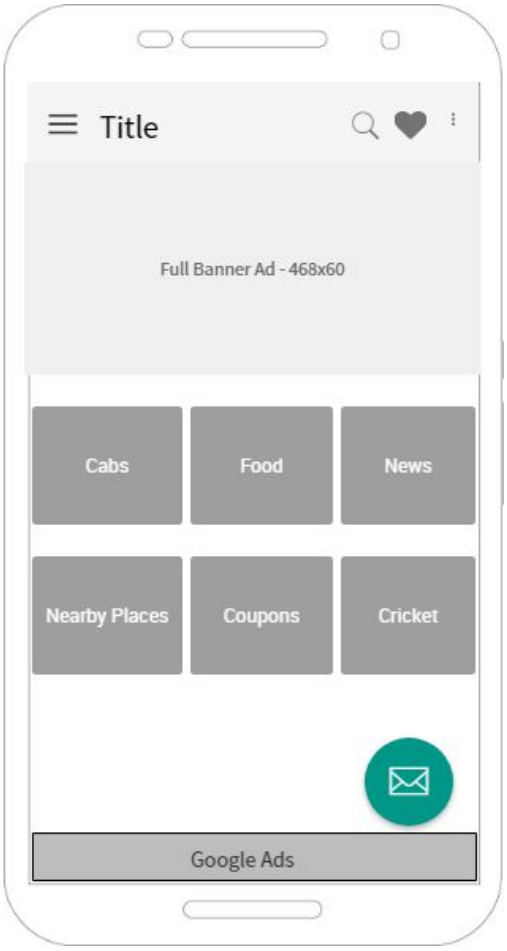
Screen 1

When the user launches the app for the very first time, it asks user to login. If the user don't have account, he/she can create an account & login into the app.

App also provides logging through Gmail and Facebook in order to make the app user friendly.

User can enter his /her login credential & click the login button for signing in.

Screen 2



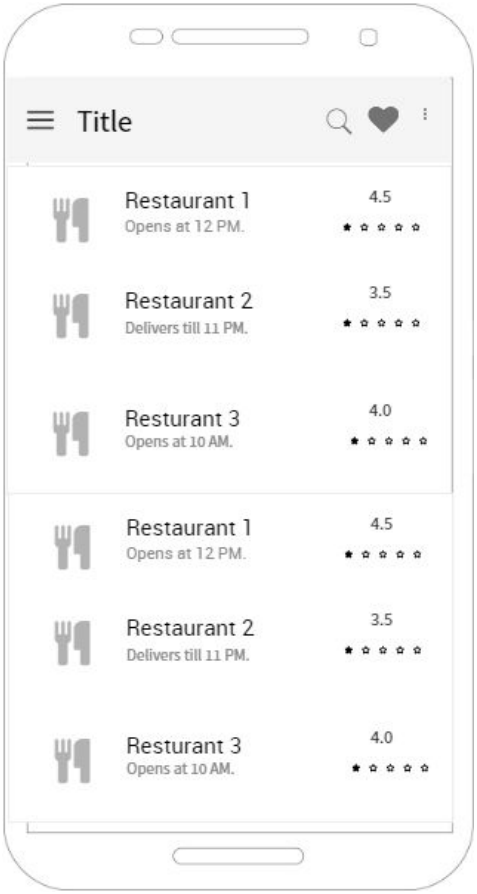
After successful login, user is navigated to main screen which consists of different menu options where the user can tap & explore a particular menu. For example:

- User can select cabs menu to book a cab nearby.
- User can select food menu to view nearby restaurant.
- User can select news menu to read news section.
- User can select nearby places menu to search nearby places.
- User can select coupons menu to view latest offers.
- User can select cricket menu to view cricket score, commentary & upcoming matches.
- User can tap on mail icon button to contact us or send us any feedback related to app.

Screen 3

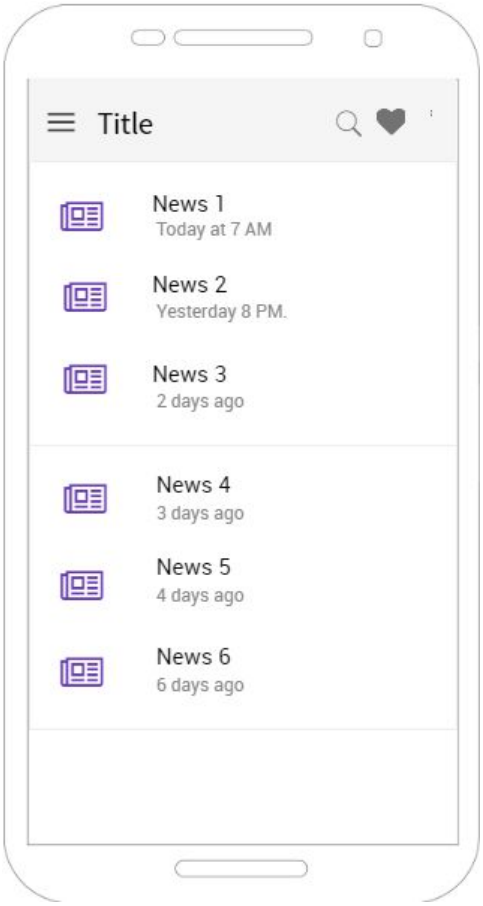
When the user selects the cab menu, it show different taxi service provider in his/her area. User can book a cab depending on the availability. User can select pick up & drop location in order to book a ride.

Screen 4



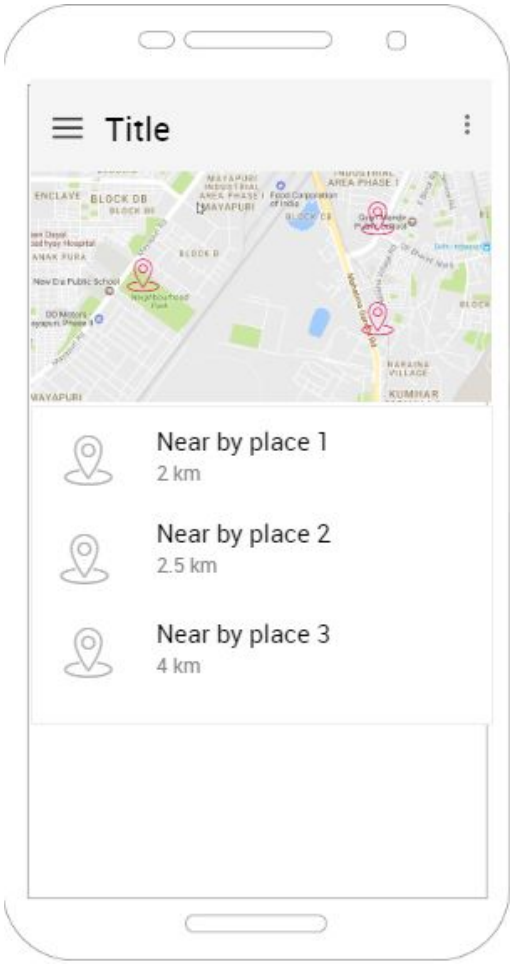
When user selects the food menu, user is navigated to screen where restaurant nearby his/her location is listed with all details & menu.

Screen 5



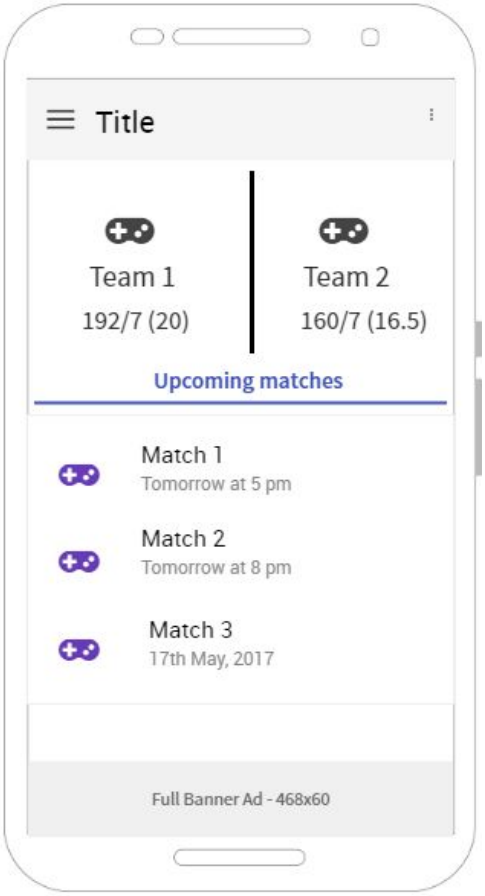
When user selects the news menu, it shows list of news which the user can read by clicking on that particular news item.

Screen 6



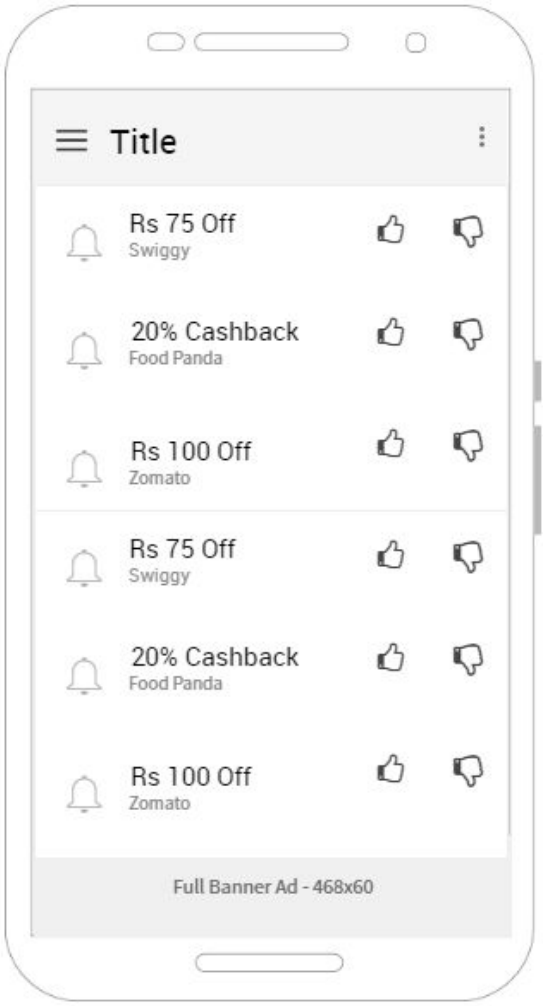
When the user selects the nearby places menu, app shows a list of nearby places to the current location of user.

Screen 7



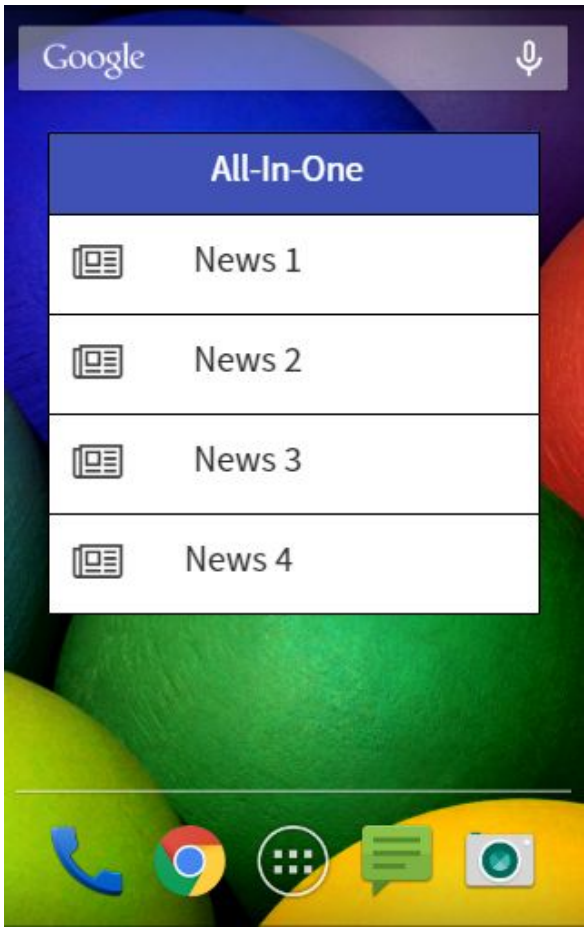
When the user clicks the cricket score menu, app shows the live score of matches & list of upcoming matches.

Screen 8



When the coupon section, it shows a list of offers available. User can select a particular offer to get coupon code or can activate particular deal.

Screen 9



UI Mockup of widget which the user can add to his/her homescreen.

Key Considerations

How will your app handle data persistence?

The app will store favourite restaurants with all their attributes and the nearby places into a local Content Provider. Any App settings will be stored with Shared Preferences.

Describe any corner cases in the UX.

- If the user tries to book a cab in an area where cab service provider does not provide service, app will show a message showing that no cab is available in your area.
- If the user tries to look for nearby restaurant, if no restaurant is available nearby his/her location, app will show that no restaurant found.
- If the user accidentally click the back button when using Google Places service to look for a Location, he will have to go back to the application and the last results will be available on the list.
- If the user rotates the device while waiting for some results from the Google Places, the system will close and will launch automatically the same request.

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

- Glide to handle the loading and caching of images.
- Retrofit for making http api calls.
- Google Places: used to find places and to use Places Picker
- Google Map: used to provide Map Views within the app
- Design Support Library: used to enhance the Material Design
- Schematic: used for automatically generating the content provider.
- Google AdMob: used for showing ads to the user.

Describe how you will implement Google Play Services.

- Google Places API will be used for search functionality i.e. places autocomplete & for getting the nearby places rich details.
- Google Maps API will be used for selecting pick up & drop off location while booking a cab.
- Google AdMob will be used for showing ads to the user.

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

- Download, configure and build any library requested by the application .
- Define all flavours of the app
- Setup gradle files to correctly import and compile the project with the libraries
- Setup the AndroidManifest.xml file
- Define activities and fragments for each view and size
- Define the services needed by the application to work smoothly.
- Define the permissions needed by the whole application
- Define the schema for database to be used for cab booking, restaurant search.
- Adding the cab booking feature using the Uber & Ola API.
- Adding the restaurant search feature using Zomato API.
- Integration of news section using Google News API.
- Integration of nearby places section using Google Places API.
- Integration of cricket score section using Cricket API.
- Integration of coupon section using CouponRani Coupon API.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for the Main Activity
- Build UI for each planned Activity/Fragment reachable from the Main Activity
- Implement Animation and shared elements with icons and fake images to setup the behavior
- Implement UI for Tablet versions

Task 3: Implement the Uber & Ola API

- Implement Google Places API & Maps API for cab booking through Uber & Ola API.
- Creation of the Content Provider and Content Resolver for the cab book section.
- Implement Contract and Helper methods
- Implement loaders

Task 4: Implement the Zomato API

- Implement Zomato API for restaurant search.
- Creation of the Content Provider and Content Resolver for the restaurant section.
- Implement Contract and Helper methods
- Implement loaders

Task 5: Implement the Google News API

- Implement News API for news section.
- Creation of the Content Provider and Content Resolver for the restaurant section.
- Implement Contract and Helper methods
- Implement loaders

Task 6: Implement the Google Places API

- Implement Google Places API for nearby places search.
- Creation of the Content Provider and Content Resolver for the restaurant section.
- Implement Contract and Helper methods
- Implement loaders

Task 7: Implement the Cricket API

- Implement Cricket API for cricket section.
- Creation of the Content Provider and Content Resolver for the restaurant section.
- Implement Contract and Helper methods
- Implement loaders

Task 8: Implement the Coupon API

- Implement Coupon API for coupon section.
- Creation of the Content Provider and Content Resolver for the restaurant section.
- Implement Contract and Helper methods
- Implement loaders

Task 9: Implement the Widget

- Create a Widget to communicate with the app for showing breaking news.

Task 10: Handle Error cases

- Test and handle error cases related to:
    - Network communication failures such as Timeout, Server errors, Network not available
    - Data errors received from the Servers.
    - Errors on User's inputs when inserting empty or unaccepted values on required fields
    - Content Provider operation errors

Task 11: Produce Signed APK

- Generate the signed version of the APK to deploy