Homework 9
Yelp Search App

Prof. Marco Papa

Developed and Designed by
Dr. Marco Papa and Rohan Madhani (TA)

Contents

1. Objectives 4
2. Background 4
   2.1 Android Studio 4
   2.2 Android 4
   2.3 Amazon Web Services (AWS) 5
   2.4 Google App Engine (GAE) 5
   2.5 Microsoft Azure 5
3. Prerequisites 6
4. High-Level Design 6
5. Implementation 7
   5.1 App Icon and Splash Screen 7
   5.2 Home screen 8
   5.3 Search Functionality 10
   5.4 Detailed Business Information Activity 11
      5.4.1 **Toolbar Options** 12
      5.4.2 **Reservation Section** 13
      5.4.3 Reservations Activity 14
   5.5 Summary of detailing and error handling 15
   5.6 Additional Info 15
6. Implementation Hints 15
   6.1 Icons 15
   6.2 Third-party libraries 15
   6.2.1 Volley HTTP requests 16
   6.2.2 Picasso 16
   6.2.3 Glide 16
   6.3 Working with action bars and menus 16
   6.4 Implementing Splash Screen 16
   6.5 Adding the App Icon 16
   6.6 Adding ellipsis to long strings 16
   6.7 Adding a button to ActionBar 17
   6.8 Implementing a Recycler view in android 17
   6.9 Adding Toasts 17

6.10 Passing variables to intent 17
6.11 Formatting Text using HTML in TextView 17
6.12 Open Link in browser 17
6.13 Back press behavior on Back button 17
6.14 SearchBar and AutoCompleteTextView 17
6.15 Implementation For Reservations 18
6.16 Implementing Dialogs 18
6.17 Sectioned RecyclerView Adapter 18
6.18 Rounded buttons/Images 19
https://stackoverflow.com/a/13872391 19
6.19 GridView 19
6.210 Using Recyclerview inside ScrollView 19
6.21 Swiping to delete feature in RecyclerView 19
6.22 Drag and Reorder feature in RecyclerView 19
6.23 Create Swipeable tabs with ViewPager2 and TabLayout 19
7. Files to Submit 20

1. Objectives

- Become familiar with Java, JSON, Android Lifecycle, and Android Studio for Android app development.
- Build a good-looking Android app.
- Learn the essentials of Google's Material design rules for designing Android apps
- Learn to use the Finnhub APIs and the Android SDK.
- Get familiar with third-party libraries like Picasso, Glide, and Volley.

The objective is to create an Android application as specified in the document below and in the video.

2. Background

2.1 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android application development, based on IntelliJ IDEA - a powerful Java IDE. On top of the capabilities you expect from IntelliJ, Android Studio offers

- Flexible Gradle - based build system.
- Build variants and multiple apk file generation.
- Code templates to help you build standard app features.
- Rich layout editor with support for drag-and-drop theme editing.
- Lint tools to catch performance, usability, version compatibility, and other problems.
- ProGuard and app-signing capabilities.
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine.

More information about Android Studio can be found at:
http://developer.android.com/tools/studio/index.html

2.2 Android

Android is a mobile operating system initially developed by Android Inc. a firm purchased by Google in 2005. Android is based on a modified version of the Linux kernel. As of Nov 2018, Android is the number 1 mobile OS, in unit sales, surpassing iOS, while iOS was still the most profitable platform.

The Official Android home page is located at:
http://www.android.com/

The Official Android Developer home page is located at:
http://developer.android.com/

2.3 Amazon Web Services (AWS)

AWS is Amazon's implementation of cloud computing. Included in AWS is Amazon Elastic Compute Cloud (EC2), which delivers scalable, pay-as-you-go compute capacity in the cloud, and AWS Elastic Beanstalk, an even easier way to quickly deploy and manage applications in the AWS cloud. You simply upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Elastic Beanstalk is built using familiar software stacks such as the Apache HTTP Server, PHP, and Python, Passenger for Ruby, IIS for .NET, and Apache Tomcat for Java.

The Amazon Web Services homepage is available at: http://aws.amazon.com/

2.4 Google App Engine (GAE)

Google App Engine applications are easy to create, easy to maintain, and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers to maintain. You simply upload your application and it's ready to go. App Engine applications automatically scale based on incoming traffic. Load balancing, micro services, authorization, SQL and NoSQL databases, memcache, traffic splitting, logging, search, versioning, roll out and roll backs, and security scanning are all supported natively and are highly customizable.

To learn more about GAE support for Node.js visit this page:

https://cloud.google.com/appengine/docs/nodejs

2.5 Microsoft Azure

The Azure cloud platform is more than 200 products and cloud services designed to help you bring new solutions to life—to solve today's challenges and create the future. Build, run, and manage applications across multiple clouds, on-premises, and at the edge, with the tools and frameworks of your choice.

To learn more about the Azure services, visit this page:
https://azure.microsoft.com/en-us/solutions/

To learn more about Azure support for Node.js visit this page:
https://docs.microsoft.com/en-us/azure/devops/pipelines/targets/webapp?view=azure-devops&tabs=yaml#deploy-a-javascript-nodejs-app

3. Prerequisites

This homework requires the use of the following components:

- Download and install Android Studio. Technically, you may use any IDE other than Android Studio such as Eclipse, but the latest SDKs may not be supported with Eclipse. We will not be providing any help on problems arising due to your choice of alternate IDEs.
- You must use the **emulator, preferably Pixel 5 with API 30**. Everything should just work out of the box.
- If you are new to Android Development, Hints are going to be your best friends!

4. High-Level Design

This homework is a mobile app version of Homework 6 and Homework 8.
In this exercise, you will develop an Android application, which allows users to search for restaurants and businesses and look at detailed information about them. Additionally, the users can schedule a reservation. Which they can review and also cancel if needed. These features among others are spread out over multiple activities and fragments. There is no hard requirement on the number of activities/fragments used as long as the behavior of the app developed is similar to the reference implementation.

This homework makes use of the backend API's that you developed as part of Assignment 8. So, you can use the same Node.js backend as Homework #8. In case you need to change something in the Node backend, make sure you do not break your Angular assignment (or deploy a separate copy) as the grading for homework will not be finished at least until 1 week later.

We suggest you use Java. Kotlin is allowed but will not be supported in piazza.

PS: This app has been designed and implemented in a Pixel 5 emulator by using SDK API 30. It is highly recommended that you use the same virtual device and API to ensure consistency.

The demo will be on an emulator using Zoom, no personal devices allowed, see the rules:

https://csci571.com/courseinfo.html#homeworks

5. Implementation

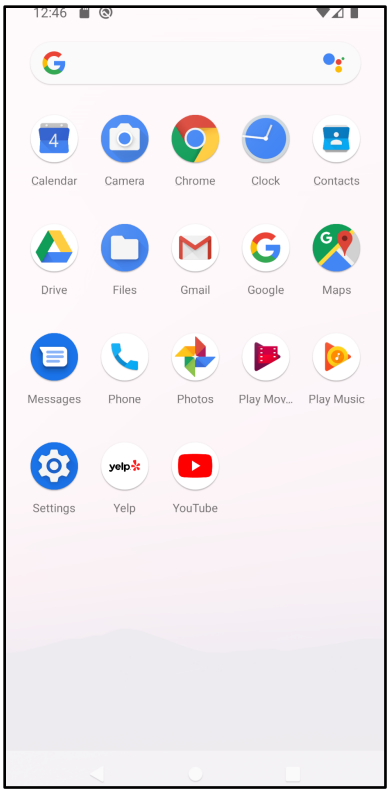


Figure 1.1
App Icon

Figure 1.2
Splash Screen

5.1 App Icon and Splash Screen

In order to get the app icon/image, please see the **hints** section. The app begins with a welcome screen (Figure 1.2) which displays the icon provided in the hint above.

This screen is called Splash Screen and can be implemented using many different methods. The simplest is to create a resource file for the launcher screen and add it as a style to AppTheme.Launcher (see **hints**). Please refer to Figure 1.1 and Figure 1.2.

5.2 Home screen

When you open the app, after being presented with the splash screen you will be shown the form which has the same elements as the previous assignments. The home screen will have a toolbar at the top with the title “Yelp” and the calendar icon.

- ***Input Form***-
  This section contains the following form field elements:
  - Keyword (text) - a required field that supports autocomplete
  - Distance (number) - a required field that accepts numeric values
  - Category (dropdown) - defaults to “Default”
  - Location (text) - Is hidden when the “auto-detect” checkbox is checked.
  - Auto Detect (checkbox)
  - Submit (button) - performs form validation and submits generating results if all elements are valid
  - Clear (button) - resets the form elements to their original values

- ***Toolbar*** -
  - “Yelp” text left aligned
  - Calendar Icon right aligned

Figure 2.1
Initial Home Screen

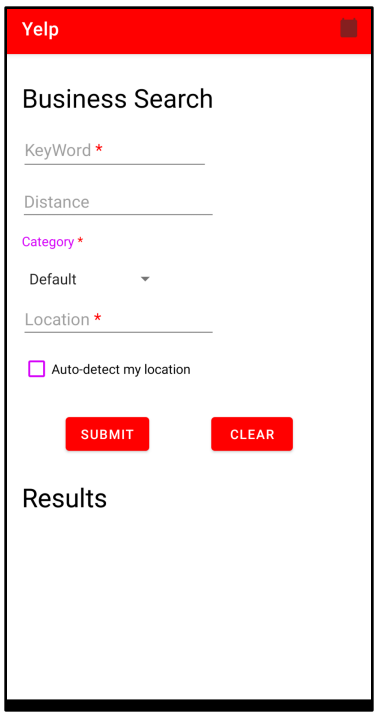


Figure 2.2
Autocomplete Options

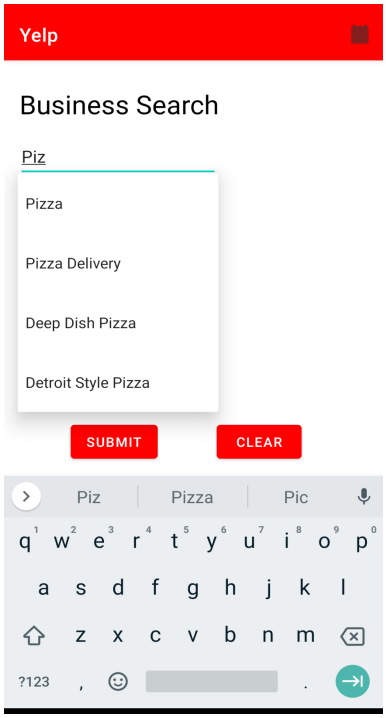
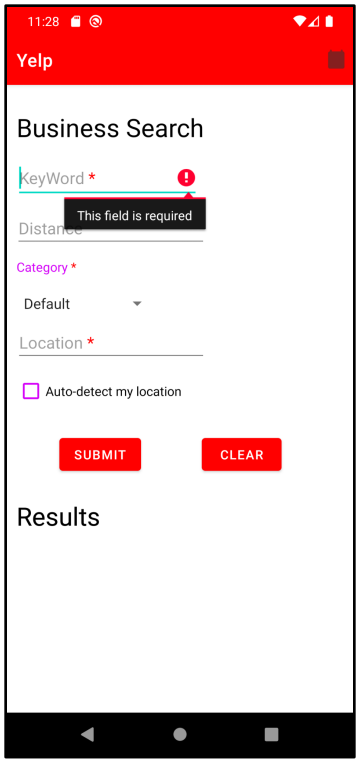


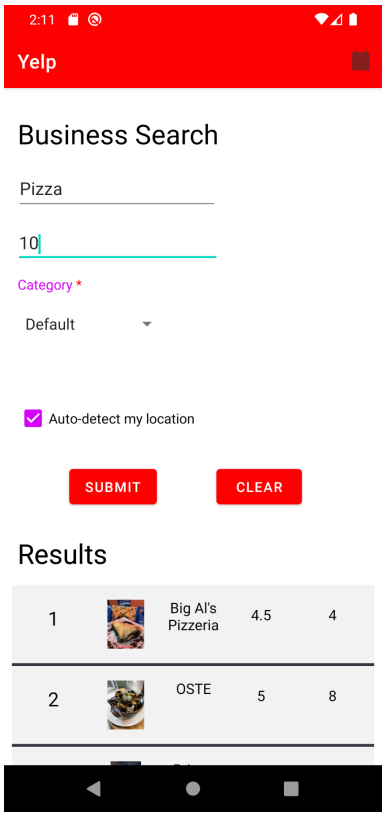
Figure 2.3
Form Validation Error



- ***Results Section*** - This section shows the results from the search that was submitted using the above form. We have five fields for each result entry, as shown below.
    - Serial Number
    - Business Image
    - Business Name
    - Rating
    - Distance

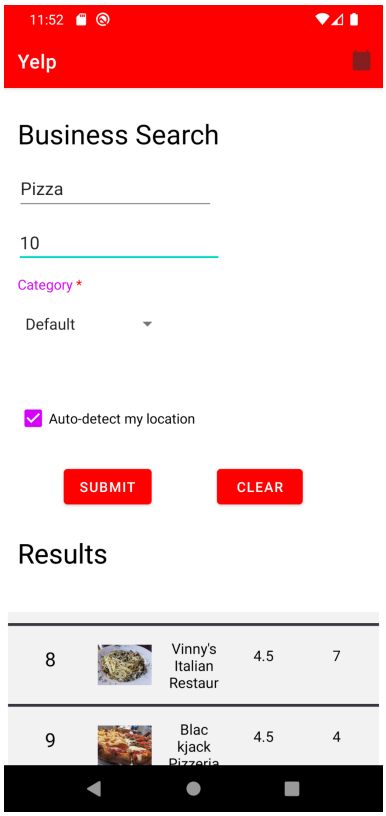
Additionally, each row in the results table is “tappable” and on tapping an entry we expect additional details about the business to be displayed using a new activity. The details of these are described below.

Figure 2.4



Initial results

Figure 2.5



Results continued

On clicking on any part of the result entry row, the detailed information screen will open for the selected business. Please refer to the hints section for the icon.

The home screen has been implemented using a Simple Activity. The home screen contains mainly two parts, the first being the form which contains TextViews, EditText, Spinner, Checkbox & Buttons. The second is the Results tab which contains the Results title and the Scrollable View where each entry is a clickable item where values are filled using the RecyclerView (can use BaseAdapter).

5.3 Search Functionality

- The form element has an input text field that supports autocomplete
- The user is provided with suggestions of keywords using the Yelp Autocomplete API. (Same as Homework #8)
- When the user taps on a suggestion, it is filled inside the text box however the search term doesn't need to be restricted to only autocomplete items.
- When the form is submitted with any of the required filed missing an alert is shown beside the respective field.
- The Category drop-down menu supports the same items as in Homework 8
- On checking the Auto-Detect location checkbox, the text input for location is hidden and behind the scenes, the current location of the device is fetched. You may use IP Info for this or play around with Android's Location Services API.
- If the checkbox is not checked, it is expected that the user provides a location as an input string that is used as part of the search query - same as we did in Homework 8.

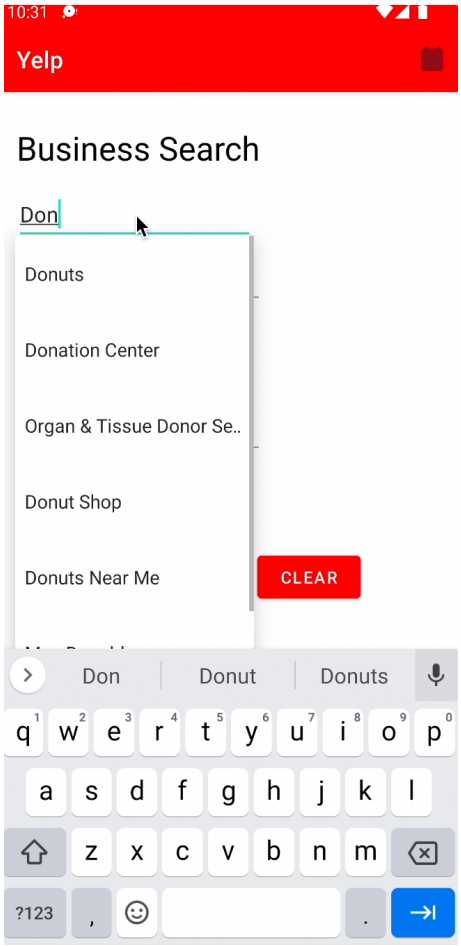


Figure 3.1
Autocomplete

5.4 Detailed Business Information Activity

On tapping on an entry from the results table we switch to a new tabbed activity that gives us more information about the selected business.

The three tabs are:

- Business Details
- Map Location
- Reviews

The Business Details tab includes in following fields about the business:

- Address
- Price Range
- Phone Number
- Status (Open / Closed)
- Category
- Url

For instructions on how to populate these fields please refer to the Homework 8 specifiaction.

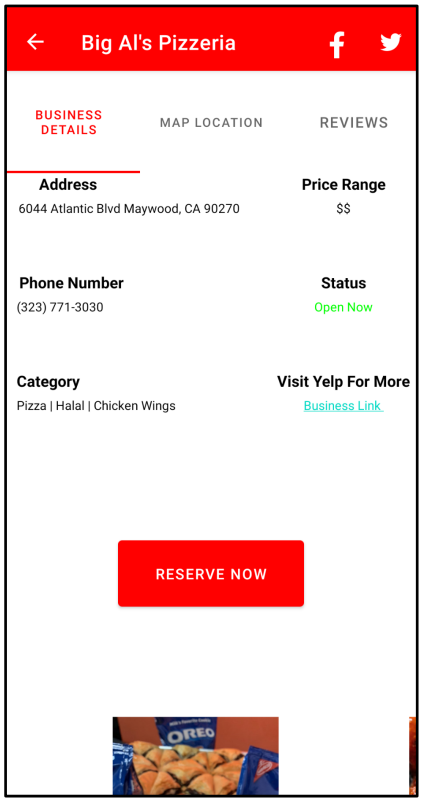


Figure 4.1
Business Details

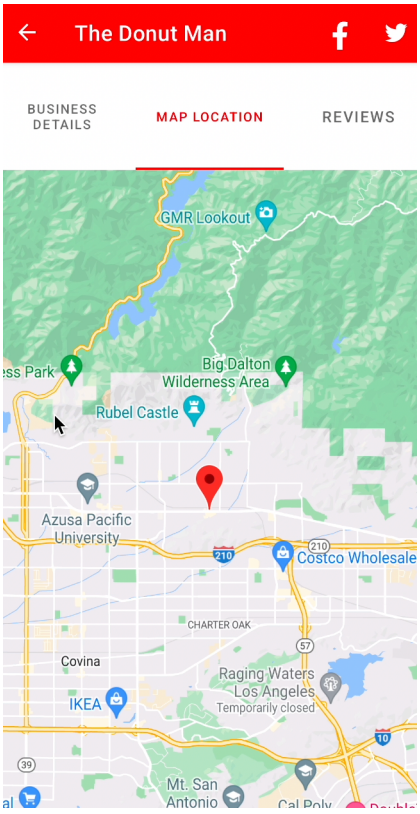


Figure 4.2
Map Location

Figure 4.3
Business Reviews

5.4.1 **Toolbar Options**

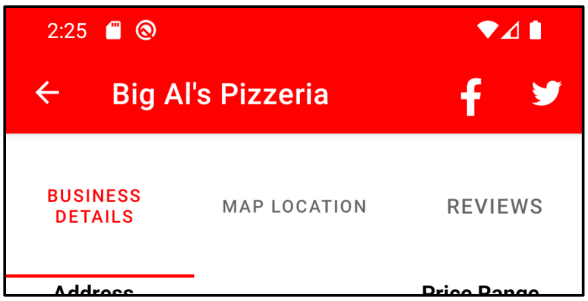


Figure 4.2
Toolbar Options

The toolbar of the business details activity changes based on the business that was selected from the results section.
The toolbar has the following sections

- Back button - Goes back to the results activity
- Business Name text field
- Share to Facebook Button
- Share to Twitter Button

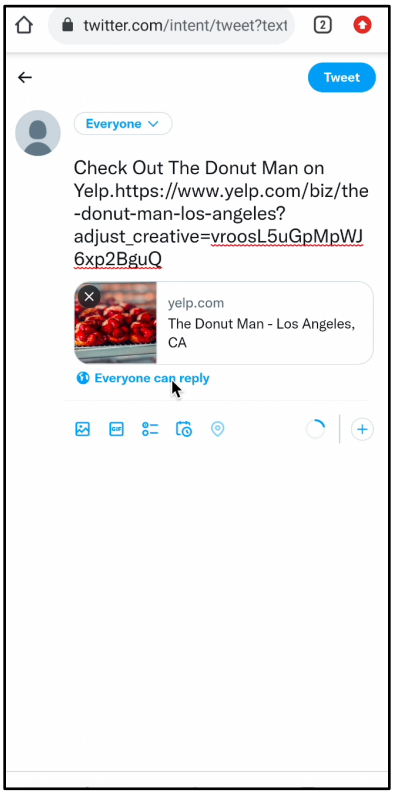


Figure 4.3 - Share to Twitter

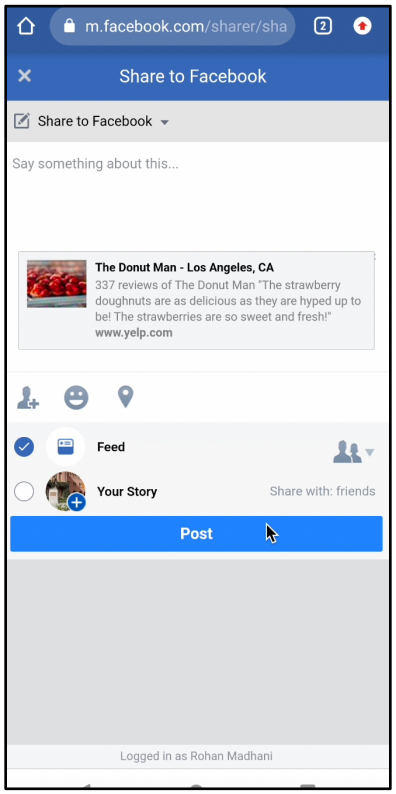


Figure 4.4 - Share to Facebook

5.4.2 **Reservation Section**

We can make a reservation at each business for a given date and time. In order to make a reservation the user must provide a valid email address. On clicking on the “Reserve Now” button we are presented with a modal that accepts the email address of the user, and the date and time for the reservation. Once the fields are validated, this information is stored in the app (Shared Preferences) and is displayed in the Bookings Activity.

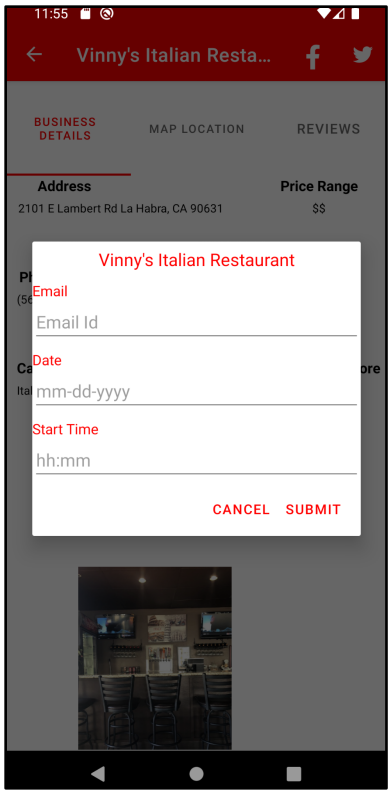


Figure 4.5
Reserve Dialog

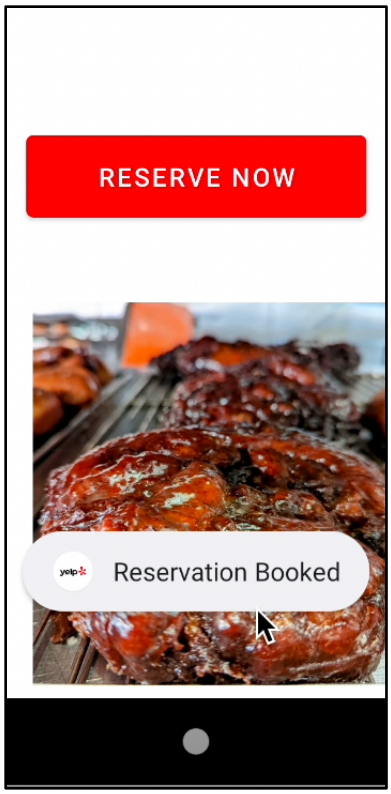


Figure 4.6
Toast on Success

After the user clicks on the submit button, a toast message is displayed on the screen depending on if the reservation was successful or not.

An attempt for making a reservation at a business would fail under the following conditions

1) If the email address is invalid
2) If the time entered is not between 10 AM to 5 PM.

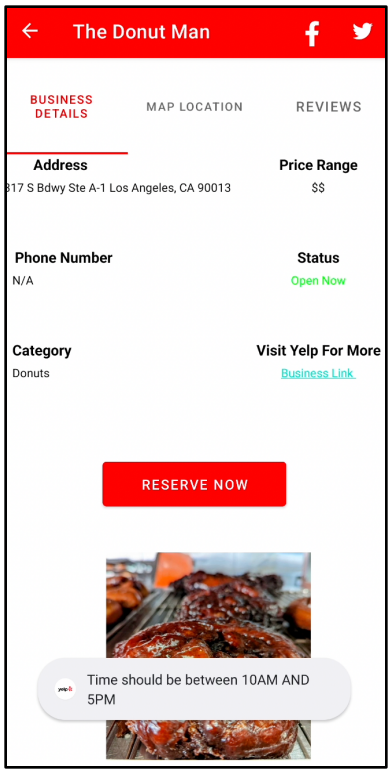


Figure 4.7
Toast on incorrect time

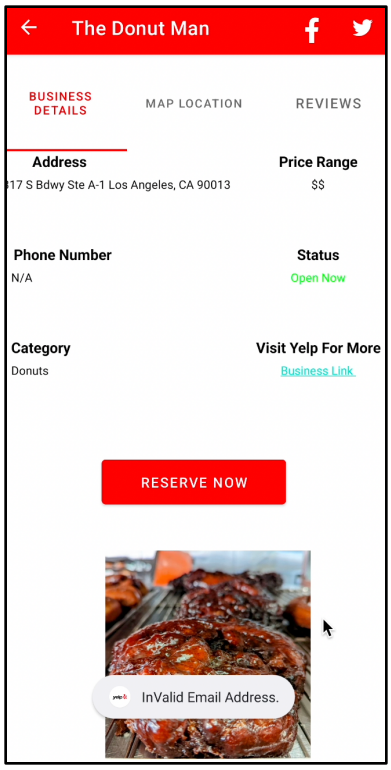


Figure 4.8
Toast on invalid email

5.4.3 Reservations Activity

From the main activity by clicking the calendar icon on the toolbar we are navigated to a new activity that shows any existing reservations that were previously made using the application. We can delete a reservation by swiping left on the entry. Upon deletion, a message is displayed confirming the deletion. You should save the list of reservations to local storage. When the app is opened, you should load them, then fetch data from your backend. We will be using Shared Preferences for storing the app data for this assignment. See hints for more details on the implementation of the Shared Preferences.

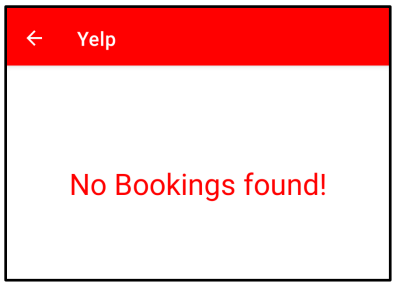


Figure 4.9
No entries in reservation

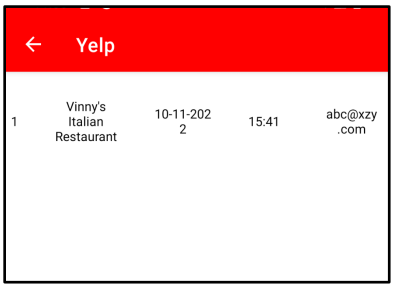


Figure 4.10
Entries exist for reservation

5.5 Summary of detailing and error handling

1. Make sure there are no conditions under which the app crashes
2. Make sure all icons and texts are correctly positioned as in the video/screenshots
3. Make sure the screens and toasts are correctly displayed.
4. Make sure the styling for different features matches the video/screenshots.
5. All API calls are to be made using Node.js backend.

5.6 Additional Info

For things not specified in the document, grading guidelines, the video, or in Piazza, you can make your own decisions. But keep in mind about the following points:

- Always display a proper message and do not crash if an error happens.
- All HTTP requests should be asynchronous and should not block the main UI thread. You can use third party libraries like Volley to achieve this in a simple manner.

6. Implementation Hints

6.1 Icons

The images used in this homework are available in the zip file mentioned in the Piazza post for Homework #9.
Furthermore, please refer to the following websites and search for additional icons.

1. https://materialdesignicons.com
2. https://icons8.com/icons/

Icon Description	**Icon name to search for**
Facebook icon	Facebook
Twitter icon	Twitter
Calendar icon	Chrome

You can choose to work with XML/png/svg/jpg versions. We recommend using XML as it is easy to modify colors by setting the Fill Colors.

6.2 Third-party libraries

Sometimes using 3rd party libraries can make your implementation much easier and quicker. Some libraries you may have to use are:

6.2.1 Volley HTTP requests

Volley can be helpful with asynchronous http request to load data. You can also use Volley network ImageView to load photos in Google tab. You can learn more about them here:
https://developer.android.com/training/volley/index.html

6.2.2 Picasso

Picasso is a powerful image downloading and caching library for Android.
http://square.github.io/picasso/

6.2.3 Glide

Glide is also a powerful image downloading and caching library for Android. It is similar to Picasso. You can also use Glide to load images..
https://bumptech.github.io/glide/

6.3 Working with action bars and menus

https://developer.android.com/training/appbar/setting-up
https://stackoverflow.com/questions/38195522/what-is-oncreateoptionsmenumenu-menu

6.4 Implementing Splash Screen

There are many ways to implement a splash screen. This blog highlights almost all of them with examples:
https://android.jlelse.eu/the-complete-android-splash-screen-guide-c7db82bce565

6.5 Adding the App Icon

https://dev.to/sfarias051/how-to-create-adaptive-icons-for-android-using-android-studio-459h

6.6 Adding ellipsis to long strings

https://stackoverflow.com/questions/6393487/how-can-i-show-ellipses-on-my-textview-if-it-is-greater-than-the-1-line

6.7 Adding a button to ActionBar

https://developer.android.com/training/appbar/actions
https://stackoverflow.com/questions/12070744/add-back-button-to-action-bar
https://stackoverflow.com/questions/34110565/how-to-add-back-button-on-actionbar-in-android-studio

6.8 Implementing a Recycler view in android

https://developer.android.com/guide/topics/ui/layout/recyclerview
https://stackoverflow.com/a/40217754
https://abhiandroid.com/materialdesign/recyclerview-gridview.html

6.9 Adding Toasts

https://stackoverflow.com/questions/3500197/how-to-display-toast-in-android
https://developer.android.com/guide/topics/ui/notifiers/toasts

6.10 Passing variables to intent

https://stackoverflow.com/questions/2405120/how-to-start-an-intent-by-passing-some-parameters-to-it

6.11 Formatting Text using HTML in TextView

https://stackoverflow.com/a/27462961

6.12 Open Link in browser

https://www.tutorialkart.com/kotlin-android/android-open-url-in-browser-activity/

6.13 Back press behavior on Back button

https://stackoverflow.com/a/27807976

6.14 SearchBar and AutoCompleteTextView

To implement the search functionality, these pages will help:
https://www.youtube.com/watch?v=9OWmnYPX1uc

https://developer.android.com/guide/topics/search/search-dialog

Working with the AutoCompleteTextView to show the suggestions might be a little challenging. This tutorial goes over how it is done to help implement it.
https://www.journaldev.com/9574/android-autocompletetextview-example-tutorial
https://developer.android.com/reference/android/widget/AutoCompleteTextView
In order to link your Search Bar with autocomplete suggestions, these links might help:
https://www.dev2qa.com/android-actionbar-searchview-autocomplete-example/

6.15 Implementation For Reservations

https://www.journaldev.com/9412/android-shared-preferences-example-tutorial

6.16 Implementing Dialogs

https://mkyong.com/android/android-custom-dialog-example/
https://androidexample.com/Custom_Dialog_-_Android_Example/index.php?view=article_discription&aid=88&aaid=111
https://medium.com/@suragch/creating-a-custom-alertdialog-bae919d2e

6.17 Sectioned RecyclerView Adapter

https://github.com/luizgrp/SectionedRecyclerViewAdapter/tree/master/app/src/main/java/io/github/luizgrp/sectionedrecyclerviewadapter/demo/example1
https://github.com/luizgrp/SectionedRecyclerViewAdapter/blob/master/app/src/main/res/layout/section_ex1_header.xml
https://github.com/luizgrp/SectionedRecyclerViewAdapter/blob/master/app/src/main/res/layout/section_ex1_item.xml
https://github.com/luizgrp/SectionedRecyclerViewAdapter/blob/master/app/src/main/res/layout/fragment_ex1.xml

6.18 Rounded buttons/Images

https://stackoverflow.com/a/13872391

6.19 GridView

https://abhiandroid.com/ui/gridview#:~:text=In%20android%20GridView%20is%20a,item%20by%20clicking%20on%20it.

6.210 Using Recyclerview inside ScrollView

https://stackoverflow.com/questions/27083091/recyclerview-inside-scrollview-is-not-working

6.21 Swiping to delete feature in RecyclerView

https://www.journaldev.com/23164/android-recyclerview-swipe-to-delete-undo#code

https://github.com/xabaras/RecyclerViewSwipeDecorator

6.22 Drag and Reorder feature in RecyclerView

https://www.journaldev.com/23208/android-recyclerview-drag-and-drop

6.23 Create Swipeable tabs with ViewPager2 and TabLayout

https://medium.com/busoft/how-to-use-viewpager2-with-tablayout-in-android-eaf5b810ef7c

https://developer.android.com/guide/navigation/navigation-swipe-view-2

7. Files to Submit

You should ZIP all your source code (without image files and third-party modules) and submit the resulting ZIP file to the DEN D2L Dropbox folder.

You will also have to submit a video of your assignment demo to the same DEN D2L Dropbox folder. You must demo your submission using Zoom. Details for that how to create the video are on DEN D2L.

Demo is done on a MacBook or Windows PC using the Android simulator, and not on a physical mobile device.