**Activity: Messages**

This documentation will technically cover two Activities, but they achieve the same purpose. The Messages activity is broken into two parts: “Messages” and “Single Message.”

“Messages” is an Activity that uses a ListView and an ArrayAdapter to display a series of items that give a preview of the most recent message between drivers and passengers. This list is only populated with threads belonging to a trip the user is a part of.

“Single Message” is the Activity that appears when the user taps one of the preview items. The preview sends an Intent with the tripId, and a JSON GET Request finds a ride by its ID number, and displays the correspondence between participants like the built-in Android Messages app. This also allows the user to send a message.

Honestly, this is probably a good place to convert both Activities into Fragments, and display them one at a time using a ViewPager.

**Important:** None of these activities update automatically. The user must reopen them to view the latest messages. Nor are there any notifications if a new message is received. I wanted to try created a Service specifically for retrieving messages after a certain period of time…

**Manifest Details**

Messages is implemented with Messages.java in the Android Manifest, with its screenOrientation set to “portrait,” and a Default Intent filter.

SingleMessage is implemented with SingleMessage.java in the Manifest. It’s screenOrientation has no value, allowing the user to rotate their phone to type in their message. It also uses the Default Intent filter.

**Layout File(s)**

Messages

* Activity\_messages.xml
  + This layout contains a single ListView called “messageListView,” to display of item\_message\_preview layouts (or message preview items). This list is populated in the Messages.java activity code behind.
* Item\_message\_preview.
  + This layout shows the most recent message within a thread. A single preview item consists of a driverImage to show the send’s profile picture, a header showing the origin and destination, a footer showing the new message, and the time the message was received. There is also a green circle that should appear when a message hasn’t been read, but that functionality has not been implemented, yet.
  + Clicking the layout will send the user to the corresponding message thread, but this click event is handled in the code behind.

Single Message

* Activity\_single\_message.xml
  + This layout displays a single thread of messages. It contains a ListView for displaying message items, and a panel for the user to type in a message. When the activity starts, it retrieves the tripId sent with the Intent from Messages, and grabs the ride’s array of messages. The ListView is then populated with a series of item\_messages\_outgoing or item\_messages\_incoming layouts, depending on the sender’s userId.
  + sendButton (TextView)
    - onClick: sendMessage – Takes the text from inputText and send the message using a JSON POST Request.
* Item\_message\_outgoing.xml
  + The message item displayed when the user sends a message.
  + messageTextView(TextView) – Contains the actual message. Its’ background property is set to @drawable/message\_outgoing\_bubble to display a green speech bubble.
  + timeTextView(TextView) – displays the time at which the message was sent.
* Item\_message\_incoming.xml
  + The message item displayed when viewing messages from the driver or other passengers.
  + userNameTextView(TextView) – Displays the sender’s name
  + messageTextView(TextView) – Displays the actual message. It’s background property is set to @drawable/message\_incoming\_bubble to display a purple speech bubble
  + timeTextView – displays the time the message was sent.

|  |  |
| --- | --- |
|  |  |

**Class File(s)**

Messages.java

* Implements Messages. This file utilizes an ArrayAdapter to populate the layout’s listView with a series of message preview items.
* Variables
  + –prefs: SharedPreferences – Grabs the user’s access token and userId
  + –userId: String – The userId is sent in the request to get the user’s trips.
* Methods and Classes
  + \*onCreate(savedInstanceState): void
    - Displays the activity\_messages layout, initializes prefs and userId, creates a back button in the app’s toolbar, and calls the getAllMessages method.
  + +onOptionsItemSelected(item): Boolean
    - Return the user to the previous screen when the back button is tapped.
  + –getAllMessages(): void
    - This retrieves a trip’s messages, and stores them into an ArrayList of HashMaps to create a message preview item.
    - Build the url and create a new RequestQueue for sending JSON request,
    - Create a new JSON Array GET Request for the user’s trips, and when the server responds, loop through the array of trips. In each iteration, get a JSON Object of the current Ride, a JSON Array of the ride’s messages, and a JSON Object of the current methods. If the Array of message is empty, skip this iteration.
    - Create a new HashMap and store the tripId, the sender’s profile picture, a header containing the origin and destination cities, footer with the actual message, and the time epoch. Add the HashMap to the ArrayList
    - After the loop, create a new instance of the ArrayAdapter, passing the ArrayList into the constructor, to create a message preview item, and add it to the ListView.
  + –MessageAdapter
    - Creates message preview items to display in the layout’s ListView
    - Variables
      * -messages: ArrayList<HashMap> -- stores the ArrayList sent from getAllMessages()
    - Methods
      * MessageAdapter(context, messages)
        + Stores the ArrayList into this.messages
      * +getView(position, convertView, parent): View
        + Inflates an item\_message\_preview layout
        + Pulls a single HashMap from the ArrayList and sets the layout’s Views to the appropriate values,
        + Creates an onClickListener for the layout, to open the SingleMessage Activity, passing the tripId in the Intent
        + Returns the Message Preview

SingleMessage.java

* Implements Single Message like Message, it utilizes an ArrayAdapter to display a chain of messages in a ListView
* Variables
  + –prefs: SharedPreferences – Gets the user’s Id and access token stored on the phone
  + –queue: RequestQueue – a queue to send JSON Requests
* Methods and Classes
  + \*onCreate(savedInstanceState): void
    - Displays the activity\_single\_message layout. Initializes queue and prefs, creates a back button in the toolbar, and calls the getConversation() method.
  + +onOptionsItemSelected(item): Boolean
    - Returns the user to the previous screen if the back button is tapped.
  + –getCoversation(): void
    - Get the current trip’s chain of messages and display them with a item\_message\_outgoing layout or an item\_message\_incoming layout, depending on the user.
    - Create a new ArrayList of HashMaps to store messages, creates a string to store the tripId from the Intent’s String Extra, and uses the tripId to build the request url.
    - Create a new JSON Object GET Request to get a trip. Create a new JSON Array that store’s the trip’s messages array, and loop through the array. In each iteration, get a single message, and create a HashMap to store the messages userId, first and last names, comment, and time sent. Add the HashMap to the arrayList.
    - After the loop, create an instance of the ArrayAdapter to display the message items in the ListView, passing the ArrayList to the constructor. Lastly, set the ListView to the list of messages.
  + +sendMessage(view): void
    - Send the text stored in the TextInputEditText. If inputText is empty, return and do nothing.
    - Create a new HashMap, pull the tripId from the Intent, pull the user’s Id from the phone, and create request’s url, using tripId and userId. Store the tripId, userId, and inputText into the HashMap.
    - Create a JSON Object POST request, passing the HashMap as a JSONObject. If the request is successful, the activity reloads, updating the ListView with the new message.
  + –MessageAdapter
    - Create message incoming or message outgoing items to display in the ListView
    - Variables
      * -messages: ArrayList<HashMap> -- stores the ArrayList sent from getConversation()
    - Methods
      * MessageAdapter(context, messages)
        + Stores the ArrayList in this.messages
      * +getView(position, convertView, parent)
        + Takes a single HashMap from the ArrayList to retrieve a message’s information.
        + Retrieve the userId from the phone. If the userId matches the message’s userId, inflate an item\_message\_outgoing layout. Else, inflate an item\_message\_incoming layout.
        + Also, if an incoming layout is inflated, set the layout’s name TextView to show the sender.
        + Show the actual message, and the time the message was sent
        + Finally, return the message item.