

HW4 – Internet and Location

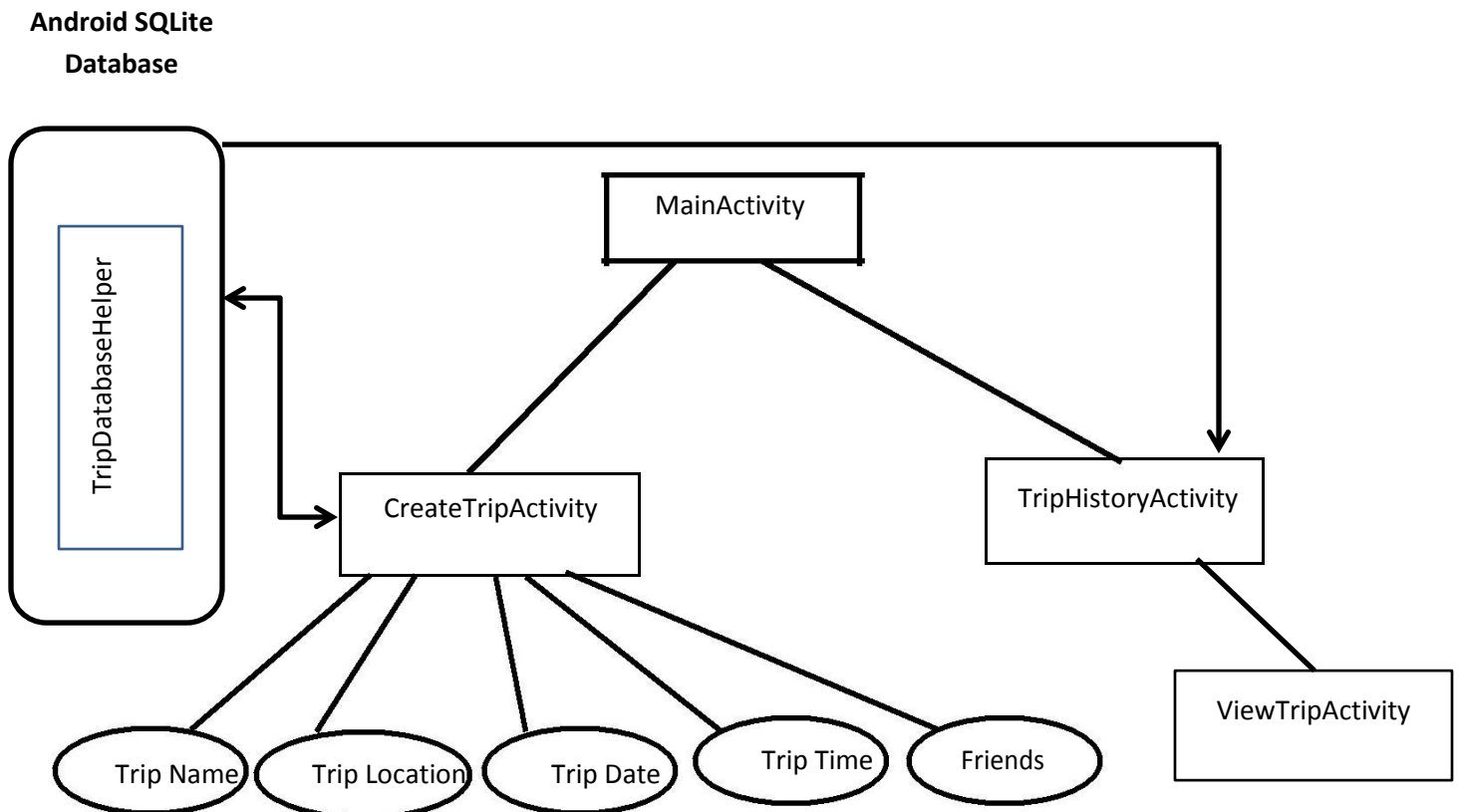
Taranjeet Singh
Submitted: 05/04/2015

ETA: Trip Scheduler Application

NOTE: The documentation for the features added in HW4 has been highlighted in yellow inside this comprehensive documentation for the ETA: Trip Scheduler Application.

Application Design

The application design/architecture for the ETA application using MVC framework is shown below



HW4 – Internet and Location

Summary of HW4 features

As part of features added in HW 4, **in general**, we are interacting with the Internet or the Web Server during 3 events using either an AsyncTask or an Intent Service in our ETA app as follows:

- 1) When a trip is started or marked as “current active trip” using START TRIP button on ViewTripActivity UI, that starts the LocationUpdateService (a customized IntentService) and triggers it periodically after every 1 minute until a trip is stopped or marked as “inactive trip” using STOP TRIP button.
- 2) When a user tries to save a newly created trip into the database, the application retrieves a 32-bit integer ID from the Web Server via getTripIdAsyncTask (an AsyncTask) and then saves the newly created trip into the database using the returned ID as the trip ID.
- 3) When a user requests the status of each person related to the “current active trip” by clicking ACTIVE TRIP STATUS button on MainActivity UI, the application retrieves the status information from the Web Server via getTripStatusAsyncTask (an AsyncTask) and displays it to the user.

Activities (Controllers)

1) MainActivity

This activity class is the home/launcher activity of the ETA app. It includes the options to ‘Schedule Trip’, ‘View Scheduled Trips’, and ‘Active Trip Status’.

The ‘Schedule Trip’ button takes the user to the CreateTripActivity which is used to create/schedule a new trip.

The ‘View Scheduled Trip’ button takes the user to the TripHistoryActivity which is used to list all the trips stored in the SQLite database. An error message is displayed in an alert dialog box if there are no scheduled trips stored in the database yet.

The ‘Active Trip Status’ button calls the getTripStatusAsyncTask to get the status of "current active trip" from the Web Server and display it in a user friendly format via a Toast. An error message is displayed in an alert dialog box if there is no “active” trip present in the database yet.

It contains the following implemented inner classes & methods:

getTripStatusAsyncTask: This private inner class represents an AsyncTask which handles the task of sending the "TRIP_STATUS" JSON request to the Web Server in a background thread to get the status/information of each person related to the "current active trip" in response.

getResponseString(): This is a method of getTripStatusAsyncTask inner class that sends the "TRIP_STATUS" JSON request to the Web Server to get the JSON response string containing the information related to "current active trip".

startCreateTripActivity(): This method starts the CreateTripActivity responsible for creating/scheduling a new Trip.

startTripHistoryActivity(): This method starts the TripHistoryActivity responsible for viewing the list of all scheduled trips stored within the database.

HW4 – Internet and Location

2) CreateTripActivity

This activity class is responsible for handling the creation/scheduling of a new trip by accepting and validating the trip details entered by the user on the screen and stores the trip into the Android SQLite database.

It uses implicit intents to access data from other applications like adding more than one friends to invite for the trip using 'Contacts' application and getting a trip location from the 'HW3API' application based on the search criteria entered by the user in the form of 'Search Area' and 'Search Item' text fields.

It contains the options to 'Save Trip' and 'Cancel Trip'. The 'Save Trip' button is enabled only after all the mandatory details are entered on the UI. It calls the `getTripIdAsyncTask` to get a trip ID for the newly created trip object from the Web Server and then saves the newly created trip object (including trip friends & trip location details) into the SQLite database using the `saveTrip()` method (called by `getTripIdAsyncTask`).

The 'Cancel Trip' button saves nothing, destroys the `CreateTripActivity` and takes the user back to the `MainActivity`.

It contains the following implemented inner classes & methods:

`DatePickerDialogFragment`: This inner class represents a reusable Date Picker hosted in a Dialog Fragment for 'ETA' app.

`onDateSet()`: This method sets the date selected from the Date Picker Dialog into the 'Trip Date' TextView.

`getTripIdAsyncTask`: This private inner class represents an `AsyncTask` which handles the task of sending the "CREATE_TRIP" JSON request to the Web Server in a background thread to get a trip ID for the newly created trip in response and then saves the newly created trip object into the SQLite database using the `saveTrip()` method of `CreateTripActivity`.

`getResponseString()`: This is a method of `getTripIdAsyncTask` inner class that sends the "CREATE_TRIP" JSON request to the Web Server to get the JSON response string containing the trip ID.

`showDatePickerDialog()`: This method displays a Date Picker Dialog Fragment to select a date on click of 'Trip Date' TextView.

`pickContact()`: This method is used to add a friend by sending an implicit intent to pick a contact name from the Phone's 'Contact Book' App.

`searchLocation()`: This method is used to populate the trip location by sending an implicit intent to search for a location (as per the search criteria entered by the user) from the 'HW3API' location finding App.

`onActivityResult()`: This method is used to get the result intents back from Phone's 'Contact Book' App as well as 'HW3API' location finding App and display the returned data into appropriate views on 'CreateTripActivity' UI.

`createTrip()`: This method is used to instantiate a Trip model object from the details entered by the user on the 'CreateTripActivity' UI.

HW4 – Internet and Location

saveTrip(): For HW4, this method will store a trip into the SQLite database by inserting the trip details and the corresponding trip friends and trip location details into the respective database tables.

cancelTrip(): This method is used when the user wants to cancel the creation of a trip.

3) TripHistoryActivity

This activity class is responsible for showing a list of all scheduled trips stored within the database. It uses a custom CursorAdaptor to populate the ListViews containing present, future, and past trips directly from the respective cursors returned from the queries fired against the database.

This activity categorizes the trips based on trips that happened in the past, upcoming trips in the future, and trips that are occurring today under the respective labels on the UI.

When clicking on a trip within the list, it retrieves the respective trip details from the database to create a Trip object which is sent to the ViewTripActivity as a Parcelable extra via an explicit intent. The trip details are displayed within the ViewTripActivity.

It contains the option to go to 'Home' for a smoother and user friendly navigation. The 'Home' button destroys/finishes the TripHistoryActivity and takes the user back to the MainActivity.

It contains the following implemented methods:

setListViewHeight(): This method is used to set the height of any of the ListViews of present, future, and past trips, based on the number of children items displayed in it.

goHome(): This method takes the user back to the MainActivity on clicking 'Home' button.

4) ViewTripActivity

This activity class is responsible for showing the details of a trip selected from the TripHistoryActivity. It retrieves the trip details of the selected trip from the trip object received as a Parcelable extra via an explicit intent and displays them on the ViewTrip UI. It displays the list of friends related to the selected trip in a ListView on the UI.

It includes the options to 'Start Trip' and 'Stop Trip'.

HW4 – Internet and Location

The 'Start Trip' button marks the currently viewed trip as the "current active trip" and sets that trip status as "active" in the database. The click event on this button also starts the LocationUpdateService which periodically updates the Web Server with user's current location (obtained via GPS) while there is a "current active trip" in the application. Error messages are displayed in an alert dialog box if there is no internet/network availability, the GPS is not enabled, or there is already another "current active trip" in the application before attempting to start the currently viewed trip. An error message is also displayed as a Toast message if the currently viewed trip is already the "current active trip"

The 'Stop Trip' button marks the currently viewed trip as "inactive" (if it is the "current active trip") and sets this trip status as "inactive" in the database. The click event on this button also stops the LocationUpdateService since this trip is no longer "active". An error message is displayed in an alert dialog box if the currently viewed trip being stopped is not the "current active trip".

It also contains the option to go back to 'Scheduled Trips' for a smoother and user friendly navigation. The 'Scheduled Trips' button destroys/finishes the ViewTripActivity and takes the user back to the TripHistoryActivity.

It contains the following implemented methods:

isGPSEnabled(): This method to checks if the device GPS is enabled or not.

isNetworkAvailable(): This method checks if the Internet/Network connectivity is available or not.

getTrip(): This method is used to get a Trip object from the Parcelable 'extra' that was passed to ViewTripActivity via an explicit Intent.

setListViewHeight(): This method is used to set the height of the ListView of friends based on the number of children items displayed in it.

viewTrip(): This method is used to populate the ViewTrip UI with the trip details using a Trip model object.

viewScheduledTrips(): This method takes the user back to the TripHistoryActivity on clicking 'Scheduled Trips' button.

HW4 – Internet and Location

Services

1) LocationUpdateService

This is a customized *IntentService* that is responsible for handling the asynchronous task of periodically updating the Web Server with the current user location obtained via the GPS on a separate handler thread while there is a "current active trip" (set by the "Start Trip" button on 'ViewTripActivity' UI). This *IntentService* achieves this task by sending an "UPDATE_LOCATION" JSON Request to the Web Server periodically.

The reasons for creating this *IntentService* for this network intensive task are as follows:

- 1) Services perform long running tasks in the background and are not correlated to a UI; so the UI is not impacted or slowed down.
- 2) *IntentService* is a simplified version of a service that handles incoming intents in a background thread.

This *IntentService* has been chosen as the method to obtain a user's GPS location it requests a user's last known GPS location using *LocationManager* system service and *GPS_PROVIDER* as the location provider and sends it to Web Server after a time interval of 1 minute continuously while there is a "current active trip" in the application.

The time interval has been chosen as 1 minute so that this network intensive task does not drain the device battery faster, since the network connections typically use the most power.

It contains the following implemented methods:

getCurrentUserLocation(): This method gets the current location of the user's device using *LocationManager* system service and "GPS_PROVIDER" as the location provider and then saves this location in a *SharedPreferences* file on internal storage for subsequent retrieval (if required).

onHandleIntent(): This overridden method of *IntentService* handles the incoming intents in a background thread. It performs the task of sending an "UPDATE_LOCATION" JSON Request to the Web Server and getting the response back.

HW4 – Internet and Location

Adapters

1) **ListViewCursorAdapter**

This adapter class represents a custom CursorAdapter that fits in between a Cursor (data source from SQLite query) and the ListView (visual representation) and configures two aspects:

- Which layout template to inflate for an item.
- Which fields of the cursor to bind to views in the template.

This custom adapter is used to populate the ListViews that display the lists of current, future, and past scheduled trips on TripHistoryActivity UI.

It contains the following implemented *methods*:

newView(): This method is used to inflate a new view (*custom_list_item.xml*) for each list view item and return it.

bindView(): This method is used to bind the required data (*trip name and trip date, here*) from the cursor to the given view (*custom_list_item.xml*).

Models

1) **Person** (Attributes: name, currentLocation)

2) **Trip** (Attributes: tripId, name, locationDetails (ArrayList), date, time, status, friends(ArrayList))

The models have been implemented using **Parcelable objects** because Parcelable is a faster method (than other methods like Serializable) of passing objects using intents/bundles between activities in an Android application.

In Parcelable, developers write custom code for serializing and deserializing so it creates less garbage objects in comparison to Serialization. The performance of Parcelable over Serialization dramatically improves (around two times faster), because of this custom implementation.

HW4 – Internet and Location

Database

1) TripDatabaseHelper

This database helper class extends SQLiteOpenHelper class from Android SDK to implement the SQLite 'trips' database for the ETA trip scheduler app. This class supports the various fields implemented within the model objects. It includes the insertion functions to take the model objects and insert them into the database.

It implements various methods to support storage, modification and retrieval of trip details (trip location details and trip friends) for the trips created using the app.

It contains the following implemented methods:

onCreate(): This method is invoked whenever a TripDatabaseHelper object is created and is used to create the required database tables (trip, person and locDetails, here).

onUpgrade(): This method is invoked whenever the database is upgraded and requires the old database version to be upgraded and the new database version.

insertTrip(): This method is used to insert the trip details into the 'trip' table and get the ID of the inserted row.

insertPerson(): This method is used to insert a single person (friend) related to a particular trip into the 'person' table based on the ID of the trip.

insertLocationDetails(): This method is used to insert the location details related to a particular trip into the 'locDetails' table based on the ID of the trip.

getLocationDetailsFromDB(): This method is used to get the ArrayList of location details related to a single trip based on the ID of the trip stored in the database.

getTripFriendsFromDB(): This method is used to get the ArrayList of friends/persons related to a single trip based on the ID of the trip stored in the database.

getTripObjectFromDB(): This method is used to get the Trip Object from the trip details based on the trip ID of the trip stored in the database.

updateTripStatus(): This method is used to update the status of a given trip as "active" or "inactive" in the 'trip' table based on the trip ID of the trip.

getActiveTripFromDB(): This method is used to get the trip ID of the "current active trip" from the database.

HW4 – Internet and Location

Views (UI)

The views (UI of the ETA application) are shown as screenshots below:

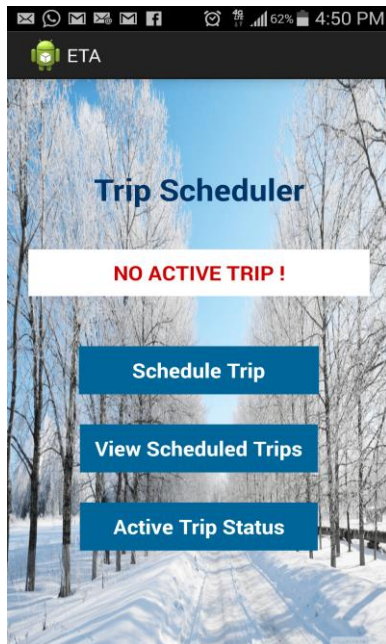


Figure 1: Main Activity View (No ACTIVE trip in the database yet)

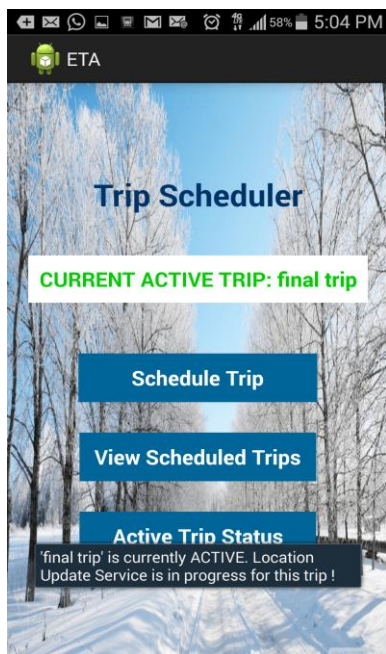


Figure 2: Main Activity View (On Application Launch: There is an ACTIVE trip in the database)

HW4 – Internet and Location

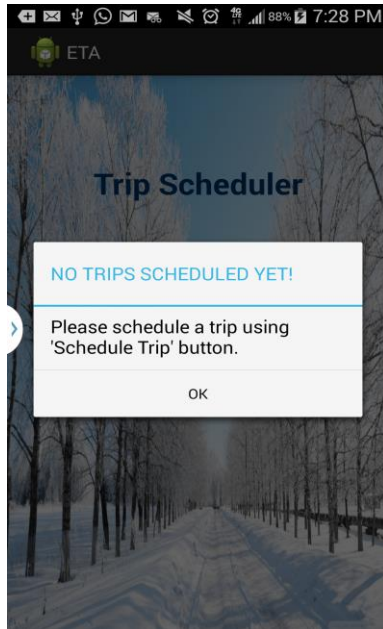


Figure 3: Main Activity View (Alert Dialog: No Scheduled Trips in Database Yet)

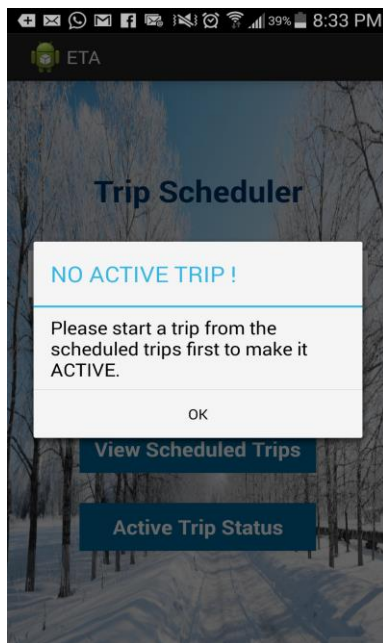


Figure 4: Main Activity View (Alert Dialog: No ACTIVE Trip in Database Yet)

HW4 – Internet and Location



Figure 5: Main Activity View (On Resume from ViewTripActivity after Starting a Trip or marking it ACTIVE)

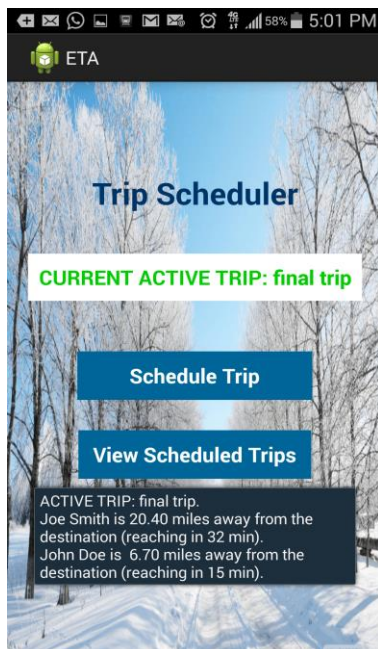


Figure 6: Main Activity View (Showing Status Information of currently ACTIVE Trip)

HW4 – Internet and Location

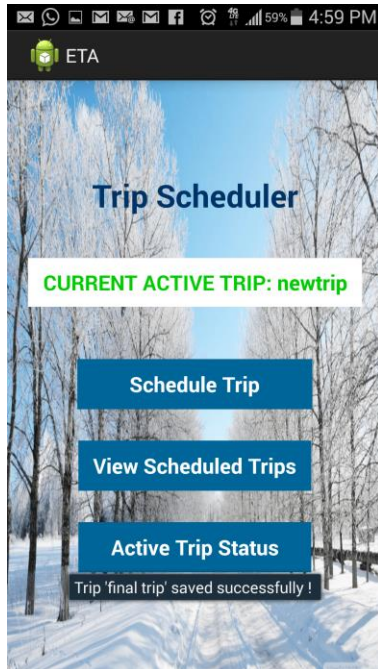


Figure 7: Main Activity View (Successful Trip Creation & Storage)

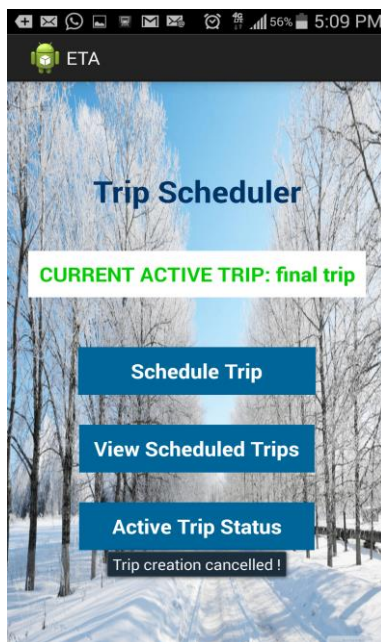


Figure 8: Main Activity View (Trip Creation Cancelled)

HW4 – Internet and Location

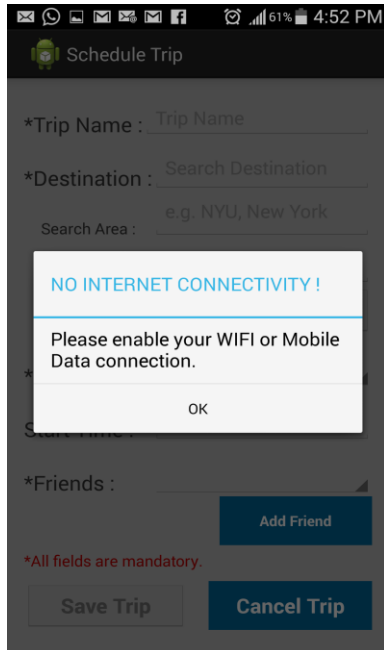


Figure 9: Create Activity View (On Activity Launch, Alert Dialog: No Internet Connectivity)

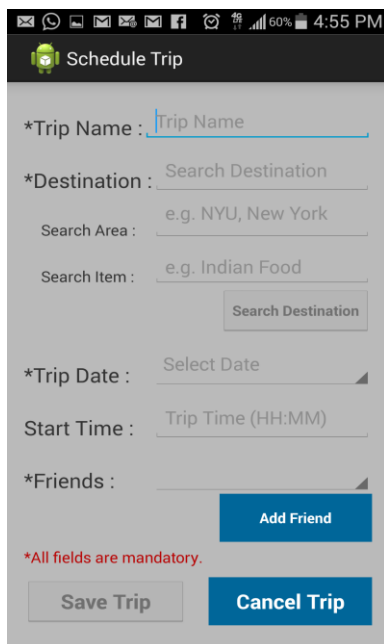
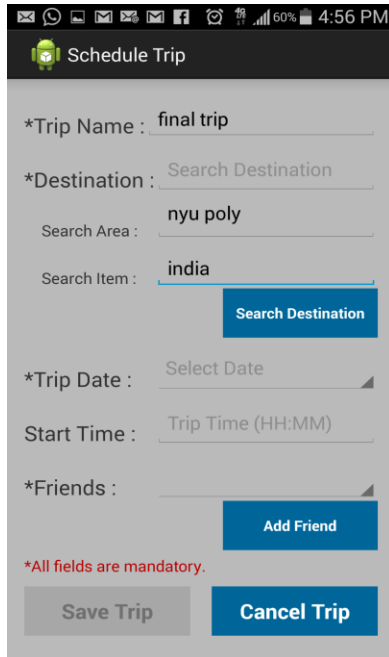


Figure 10: Create Activity View (On Activity Launch, when Internet Connectivity is Available)

HW4 – Internet and Location



The screenshot shows the 'Schedule Trip' app interface. At the top, there's a status bar with various icons and the time 4:56 PM. Below the title bar, there are several input fields: '*Trip Name : final trip', '*Destination : Search Destination', 'Search Area : nyu poly', and 'Search Item : india'. A blue 'Search Destination' button is positioned to the right of the 'Search Item' field. Below these fields are '*Trip Date : Select Date', 'Start Time : Trip Time (HH:MM)', and '*Friends :'. A blue 'Add Friend' button is to the right of the 'Friends' field. At the bottom, there are two buttons: 'Save Trip' (disabled) and 'Cancel Trip' (active). A red text note '*All fields are mandatory.' is located above the 'Save Trip' button.

Figure 11: Create Activity View (Searching for a Location using HW3API app)

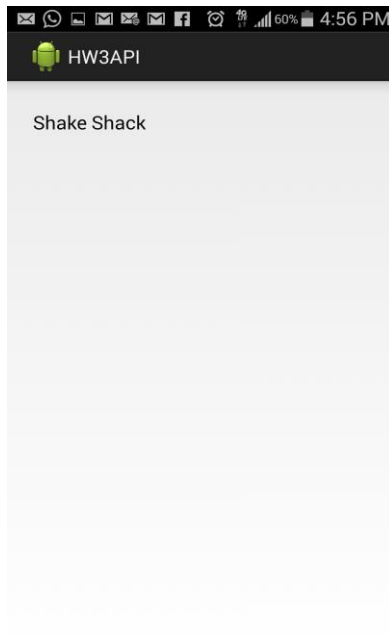
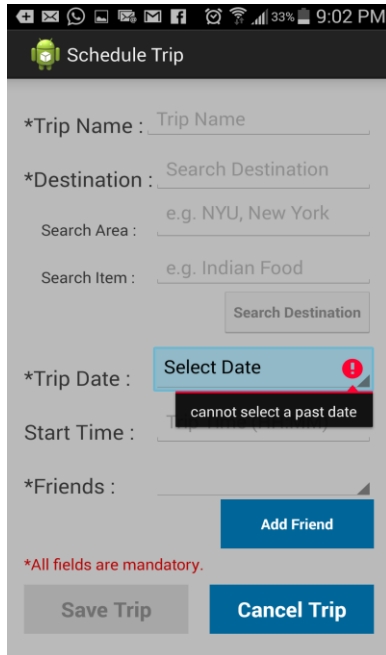


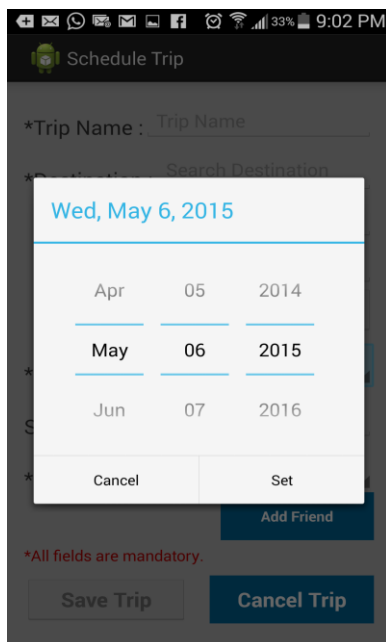
Figure 12: HW3API Activity View (Search Results returned from HW3API app)

HW4 – Internet and Location



The screenshot shows the 'Schedule Trip' app interface. At the top, there's a status bar with icons for notifications, battery, and time (9:02 PM). The app title 'Schedule Trip' is in a dark header. Below it, there are several input fields: '*Trip Name : Trip Name', '*Destination : Search Destination', 'Search Area : e.g. NYU, New York', and 'Search Item : e.g. Indian Food'. A 'Search Destination' button is next to the search item field. The '*Trip Date :' field is highlighted with a blue box, and a date picker dialog is open, showing 'Wed, May 6, 2015'. A red error message 'cannot select a past date' is displayed over the date picker. Below the date field is the 'Start Time :' field. There's also a '*Friends :' field with an 'Add Friend' button. At the bottom, there are 'Save Trip' and 'Cancel Trip' buttons. A red note '*All fields are mandatory.' is visible.

Figure 13: Create Trip Activity View (Error: On Selecting a Past Trip Date from Date Picker Dialog Fragment)



This screenshot shows the same 'Schedule Trip' app interface as Figure 13, but with a valid date selected. The date picker dialog is still open, showing 'Wed, May 6, 2015'. The error message is no longer present. The background app interface remains the same, with the same input fields and buttons.

Figure 14: Create Trip Activity View (Selecting a valid Trip Date from Date Picker Dialog Fragment)

HW4 – Internet and Location

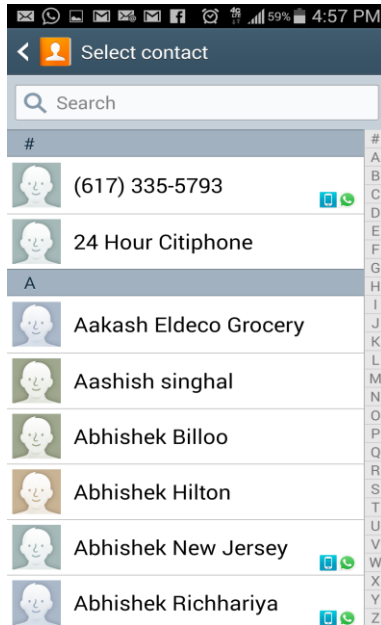


Figure 15: 'Select Contact' Activity View (Picking a Contact as a Friend from Phone's Contacts app)

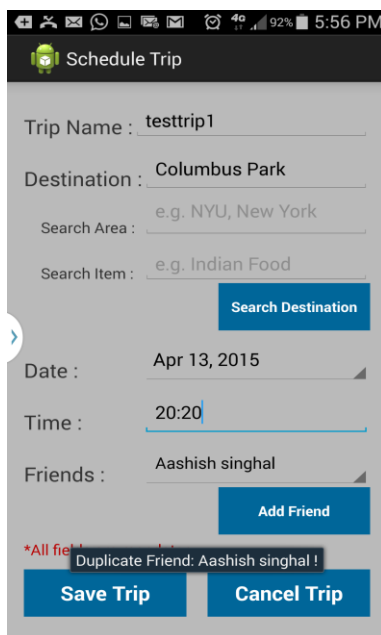
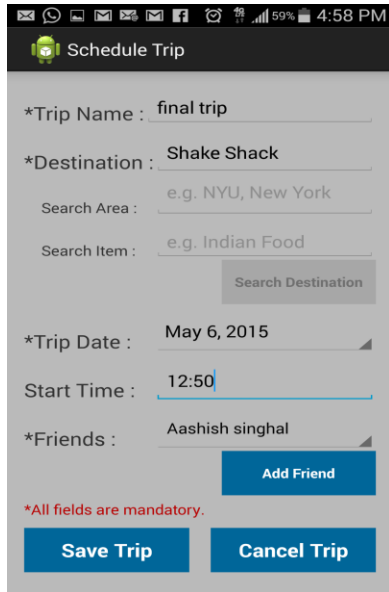


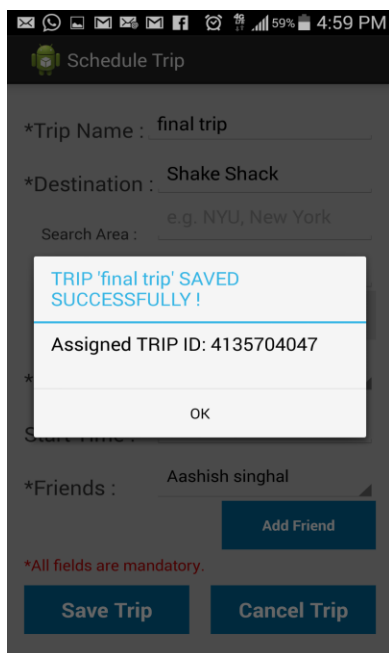
Figure 16: Create Trip Activity View (Error: Selecting Duplicate Contact as a Friend)

HW4 – Internet and Location



The screenshot shows the 'Schedule Trip' app interface. At the top, there's a status bar with icons for notifications, battery, and time (4:58 PM). The app title 'Schedule Trip' is in a dark header. Below it, the form has the following fields: '*Trip Name : final trip', '*Destination : Shake Shack', 'Search Area : e.g. NYU, New York', 'Search Item : e.g. Indian Food', '*Trip Date : May 6, 2015', 'Start Time : 12:50', and '*Friends : Aashish singhal'. There is a 'Search Destination' button next to the search fields. At the bottom, there are 'Save Trip' and 'Cancel Trip' buttons. A red note at the bottom left says '*All fields are mandatory.'.

Figure 17: Create Trip Activity View (Successfully Added Friend displayed in Spinner View)



This screenshot shows the same 'Schedule Trip' app interface as Figure 17, but with a success alert dialog overlaid. The dialog is a white box with a blue border. It contains the text 'TRIP 'final trip' SAVED SUCCESSFULLY!' in blue, followed by 'Assigned TRIP ID: 4135704047' in black. At the bottom of the dialog is an 'OK' button. The background form is dimmed, showing the same fields and buttons as before.

Figure 18: Create Trip Activity View (Trip Saved Successfully: Alert Dialog displaying Trip ID returned from Web Server)

HW4 – Internet and Location

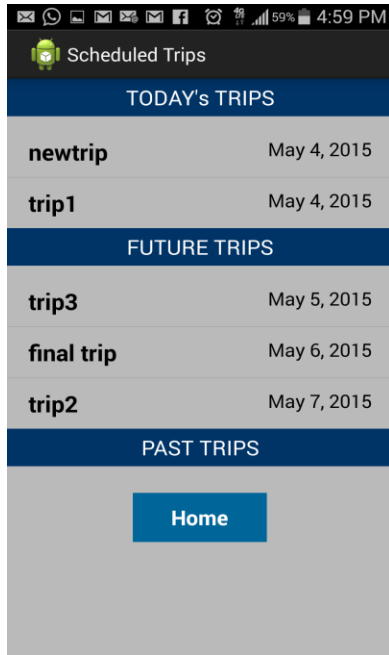


Figure 19: Trip History Activity View (Displaying Scheduled Trips from the database)

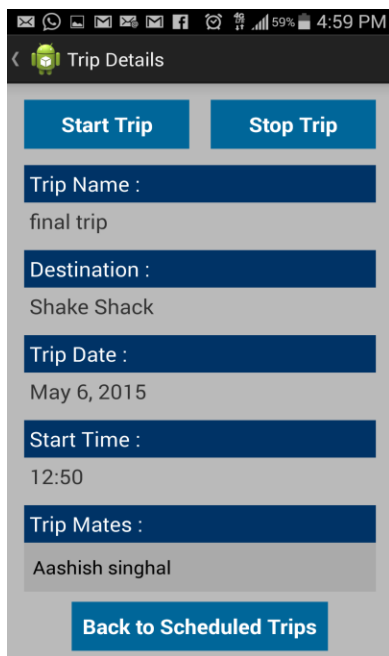


Figure 20: View Trip Activity View (On Activity Launch)

HW4 – Internet and Location

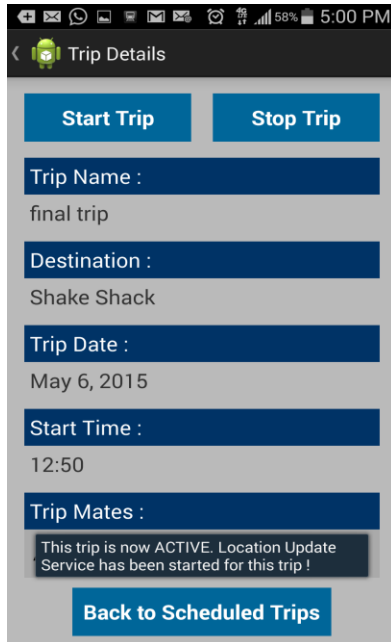


Figure 21: View Trip Activity View (Starting/Activating a Trip: Starts the LocationUpdateService for the trip)

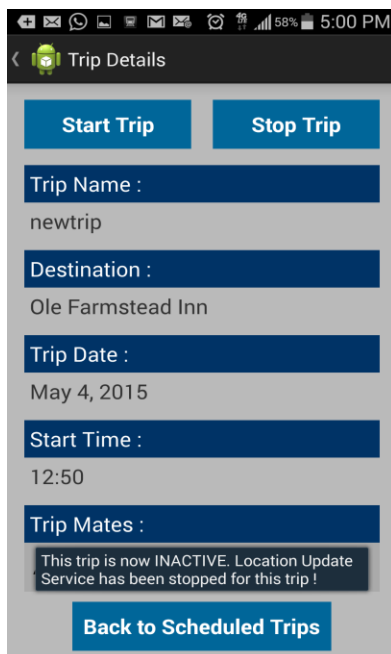


Figure 22: View Trip Activity View (Stopping an ACTIVE Trip: Stops the LocationUpdateService for the trip)

HW4 – Internet and Location

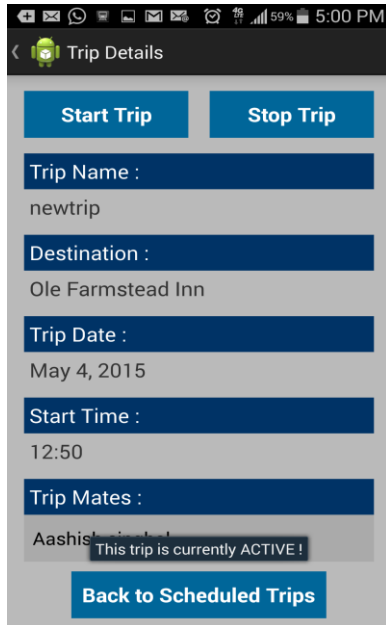


Figure 23: View Trip Activity View (Error: On Starting an already ACTIVE Trip)

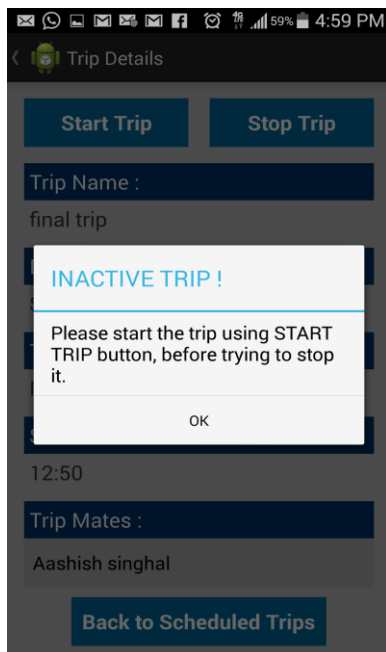


Figure 24: View Trip Activity View (Alert Dialog Error: On Stopping an already INACTIVE Trip)

HW4 – Internet and Location

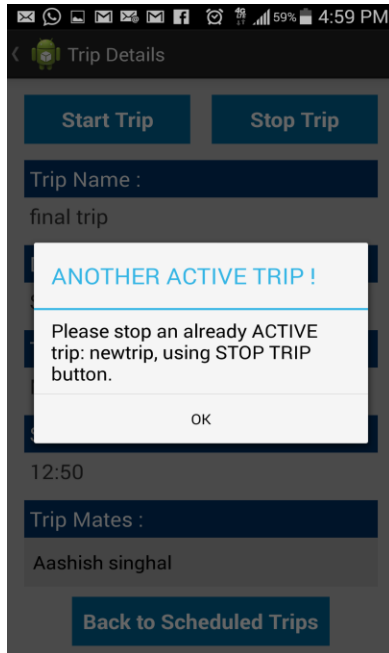


Figure 25: View Trip Activity View (Alert Dialog Error: On Starting another Trip when there is already an ACTIVE Trip)

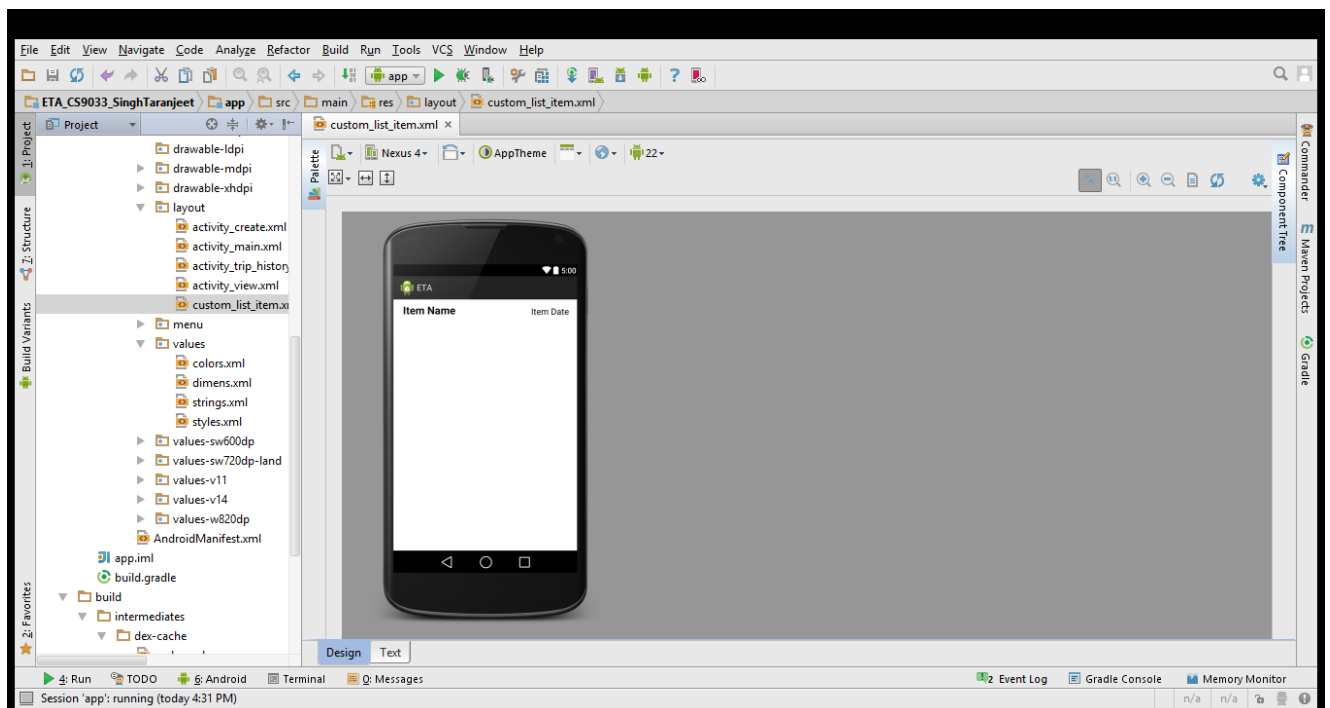


Figure 26: Custom_List_Item View (Used as a Child View for each trip item in the ListView of trips)

HW4 – Internet and Location

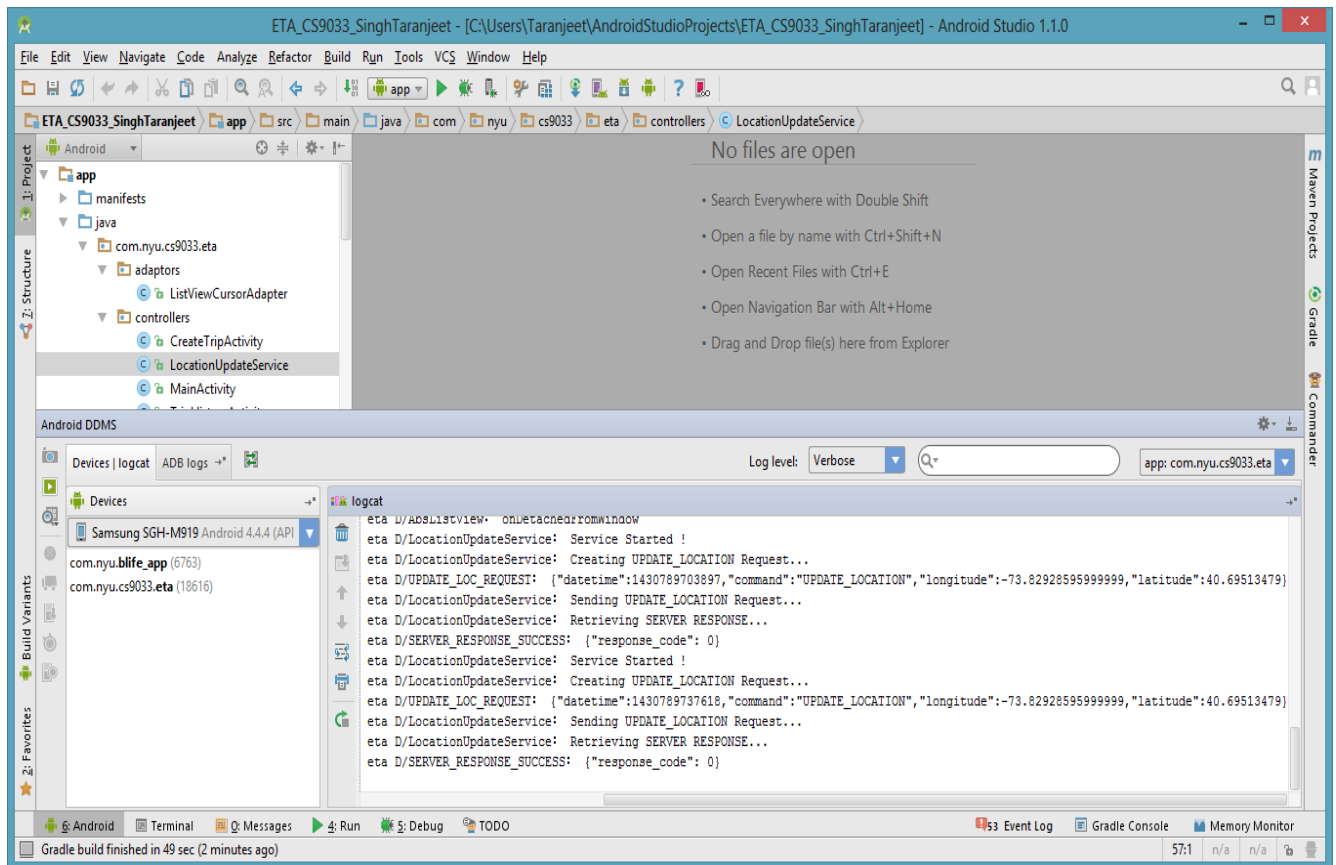


Figure 27: LocationUpdateService (IntentService) running in the background to send User GPS Location updates to the Web Server after every 1 minute for the “current active trip”