Report CS522 Assignment 9

Name: Yunfeng Wei
CWID: 10394963
E-mail: ywei14@stevens.edu
Video: In the ZIP archive file

Step:

1. Modify the Client, ClientContract, Message and MessageContract class to enable data for latitude and longitude as Figure 1.1, Figure 1.2.

```
public class ClientContract {
    public static final String CLIENT_ID = "_id";
    public static final String NAME = "name";
    public static final String UUID = "uuid";
    public static final String LONGITUDE = "longitude";
    public static final String LATITUDE = "latitude";
    public static final String ADDRESS = "address";

    public static final String TABLE_NAME = "Clients";
    public static final String CONTENT = "Client";

    public static final Uri CONTENT_URI = MessageDbProvider.CONTENT_URI(MessageDbProvider.AUTHORITY, TABLE_NAME);

    public static Uri CONTENT_URI(String id) {
        return MessageDbProvider.withExtendedPath(CONTENT_URI, id);
    }

    public static String CONTENT_PATH = MessageDbProvider.CONTENT_PATH(CONTENT_URI); // Messages
    public static String CONTENT_ITEM_PATH = MessageDbProvider.CONTENT_PATH(CONTENT_URI("#")); // Messages/#

    public static long getClientId(Uri uri) { return Long.parseLong(uri.getLastPathSegment()); }

    public static long getClientId(Cursor cursor) {
        return cursor.getLong(cursor.getColumnIndexOrThrow(CLIENT_ID));
    }

    public static void setClientId(ContentValues values, long id) { values.put(CLIENT_ID, id); }

    public static String getName(Cursor cursor) {
        return cursor.getString(cursor.getColumnIndexOrThrow(NAME));
    }

    public static void setName(ContentValues values, String name) { values.put(NAME, name); }

    public static java.util.UUID getUUID (Cursor cursor) {
        String uuid = cursor.getString(cursor.getColumnIndexOrThrow(UUID));
        if (uuid.equals("")) {
            return null;
```

Figure 1.1

```
public class MessageContract {
    public static final String MESSAGE_ID = "_id";
    public static final String CHATROOM_FK = "chatroom_fk"; // CHATROOM_FK
    public static final String MESSAGE_TEXT = "text";
    public static final String TIMESTAMP = "timestamp";
    public static final String SEQNUM = "seqnum";
    public static final String SENDER_ID = "senderID"; // SENDER_FK
    public static final String LONGITUDE = "longitude";
    public static final String LATITUDE = "latitude";

    public static final String TABLE_NAME = "Messages";
    public static final String CONTENT = "Message";

    public static final Uri CONTENT_URI = MessageDbProvider.CONTENT_URI(MessageDbProvider.AUTHORITY, TABLE_NAME);

    public static Uri CONTENT_URI(String id) {
        return MessageDbProvider.withExtendedPath(CONTENT_URI, id);
    }

    public static String CONTENT_PATH = MessageDbProvider.CONTENT_PATH(CONTENT_URI); // Messages
    public static String CONTENT_ITEM_PATH = MessageDbProvider.CONTENT_PATH(CONTENT_URI("#")); // Messages/#

    public static long getMessageId(Uri uri) { return Long.parseLong(uri.getLastPathSegment()); }

    public static long getMessageId(Cursor cursor) {
        return cursor.getLong(cursor.getColumnIndexOrThrow(MESSAGE_ID));
    }

    public static void setMessageId(ContentValues values, long id) { values.put(MESSAGE_ID, id); }

    public static long getChatroomId(Cursor cursor) {
        return cursor.getLong(cursor.getColumnIndexOrThrow(CHATROOM_FK));
    }

    public static void setChatroomId(ContentValues values, long chatroom_fk) {
        values.put(CHATROOM_FK, chatroom_fk);
    }

    public static String getMessageText(Cursor cursor) {
        return cursor.getString(cursor.getColumnIndexOrThrow(MESSAGE_TEXT));
```

Event Log Gradle Console Memory Moni

Figure 1.2

2. Modify the DatabaseHelper and MessageDbProvider as Figure 2.1 to make the database support query for latitude and longitude of clients or messages.

```
public class DatabaseHelper extends SQLiteOpenHelper {
    public static final String TAG = DatabaseHelper.class.getSimpleName();

    private static final String MESSAGE_TABLE_CREATE =
            "CREATE TABLE " + MessageContract.TABLE_NAME + " ( " + MessageContract.MESSAGE_ID + " INTEGER PRIMARY KEY AUTOI
                    MessageContract.MESSAGE_TEXT + " TEXT NOT NULL, " + MessageContract.TIMESTAMP + " INTEGER NOT NULL, " +
                    MessageContract.SEQNUM + " INTEGER NOT NULL, " +
                    MessageContract.SENDER_ID + " INTEGER NOT NULL, " +
                    MessageContract.CHATROOM_FK + " INTEGER NOT NULL, " +
                    MessageContract.LATITUDE + " TEXT NOT NULL, " +
                    MessageContract.LONGITUDE + " TEXT NOT NULL, " +
                    "FOREIGN KEY (" + MessageContract.SENDER_ID + ") REFERENCES " + ClientContract.TABLE_NAME +
                    "(" + ClientContract.CLIENT_ID + ") ON DELETE CASCADE, " +
                    "FOREIGN KEY (" + MessageContract.CHATROOM_FK + ") REFERENCES " + ChatroomContract.TABLE_NAME +
                    "(" + ChatroomContract.ID + ") ON DELETE CASCADE);";

    private static final String CLIENT_TABLE_CREATE =
            "CREATE TABLE " + ClientContract.TABLE_NAME + " ( " + ClientContract.CLIENT_ID + " INTEGER PRIMARY KEY, " +
                    ClientContract.NAME + " TEXT NOT NULL, " +
                    ClientContract.LATITUDE + " TEXT NOT NULL, " +
                    ClientContract.LONGITUDE + " TEXT NOT NULL, " +
                    ClientContract.ADDRESS + " TEXT NOT NULL, " +
                    ClientContract.UUID + " TEXT );";
```

Figure 2.1

3. In the ChatAppActivity, implement LocationListener, GoogleApiClient.ConnectionCallbacks and GoogleApiClient.OnConnectionFailedListener. First, define a GoogleApiClient and a

LocationManager to connect to Google Service and manage GPS provider as Figure 3.1. Override the onStart and onStop method to connect or disconnect to the Google Server as Figure 3.2.

```
private GoogleApiClient googleApiClient;
private LocationManager locationManager = null;

private ServiceHelper serviceHelper = null;
private AckReceiverWrapper.IReceiver receiver = null;
private AckReceiverWrapper wrapper = null;


@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    locationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
    googleApiClient = new GoogleApiClient.Builder(this).addApi(LocationServices.API)
            .addConnectionCallbacks(this).addOnConnectionFailedListener(this).build();
```

Figure 3.1

```
@Override
protected void onStart() {
    super.onStart();
    if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
        googleApiClient.connect();
    } else {
        //Toast.makeText(this, "Please enable GPS settings", Toast.LENGTH_SHORT).show();
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Please enable GPS settings first!").setPositiveButton("Setting", (dialog, which) -> {
                Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                startActivity(intent);
                dialog.cancel();
        });
        builder.create().show();
    }
}

@Override
protected void onStop() {
    googleApiClient.disconnect();
    super.onStop();
}
```

Figure 3.2

4. Implement the ConnectionCallbacks and OnConnectionFailedListener interface as Figure 4.1. The onConnected method will request location updates using GPS provider. When the connection is temporarily stopped, the location updates will be removed by the LocationManager. If the connection failed, it will cancel the connection.

```java
/* Google API Clients */
@Override
public void onConnected(Bundle bundle) {
    /* Location-based Service */
    Log.i(TAG, "Google Api Connected, try to update location");
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, UPDATE_INTERVAL, MIN_DISTANCE_IN_METERS, this
    /* End */
}

@Override
public void onConnectionSuspended(int i) { locationManager.removeUpdates(this); }

@Override
public void onConnectionFailed(ConnectionResult connectionResult) {
    if (connectionResult.hasResolution()) {
        try {
            connectionResult.startResolutionForResult(this, ConnectionResult.CANCELED);
        } catch (IntentSender.SendIntentException e) {
            e.printStackTrace();
        }
    }
}
```

Figure 4.1

5. Implement the LocationListener interface as Figure 5.1 and Figure 5.2. When the location is changed, override the client's location information and update the database.

```java
@Override
public void onLocationChanged(Location location) {
    if (latitude == 0 && longitude == 0) {
        Toast.makeText(getApplicationContext(), "Initialize location", Toast.LENGTH_SHORT).show();
    }
    latitude = location.getLatitude();
    longitude = location.getLongitude();
    Log.i(TAG, "Location changed, latitude:" + String.valueOf(latitude) +
            " longitude:" + String.valueOf(longitude));
    //Toast.makeText(getApplicationContext(), "Location Updated", Toast.LENGTH_SHORT).show();
    ClientManager manager = new ClientManager(this, (cursor) -> {
            return new Client(cursor);
    }, CHAT_APP_LOADER_ID);
    if (client != null) {
        //Log.i(TAG, "Client registered, update location");
        client.latitude = latitude;
        client.longitude = longitude;
        AddressTask task = new AddressTask();
        try {
            client.address = task.execute(location).get();
        } catch (Exception e) {
            e.printStackTrace();
        }
        //Log.i(TAG, "Get client address:" + client.address);
        manager.UpdateGPSAsync(client);
    } else {
        Log.i(TAG, "Client has not registered, do nothing");
    }
}
```

Figure 5.1

```
@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
    String str = "";
    switch (status) {
        case 0:
            str = "OUT_OF_SERVICE";
            break;
        case 1:
            str = "TEMPORARILY UNAVAILABLE";
            break;
        case 2:
            str = "AVAILABLE";
            break;
        default:
            str = "";
    }
    Log.i(TAG, "Provider:" + provider + " status changed to: " + str);
}

@Override
public void onProviderEnabled(String provider) {
    Log.i(TAG, "Provider: " + provider + " is enabled");
}

@Override
public void onProviderDisabled(String provider) {
    Log.i(TAG, "Provider: " + provider + " is disabled");
}
```

Figure 5.2

6. In the RequestService class, define a new method called getAddress as Figure 6.1. The method uses Geocoder to get the address by latitude and longitude data.

```
private String getAddress(final Client client) {
    if (client.latitude != 0 && client.longitude != 0) {
        Geocoder geocoder = new Geocoder(this, Locale.getDefault());
        List<Address> addresses = null;
        try {
            addresses = geocoder.getFromLocation(client.latitude, client.longitude, 1);
        } catch (IOException e) {
            e.printStackTrace();
        }
        if (addresses != null) {
            Address address = addresses.get(0);
            String addr = "";
            for (int i = 0; i < address.getMaxAddressLineIndex(); i++) {
                addr += address.getAddressLine(i);
                addr += "\n";
            }
            return addr;
        } else {
            return "";
        }
    } else {
        return "";
    }

}
```

Figure 6.1

7. Modify the layout of messages and clients to show the latitude, longitude information or address information on the screen as Figure 7.1 and Figure 7.2.

```
String[] from = new String[] {MessageContract.MESSAGE_TEXT, ClientContract.NAME, MessageContract.LATITUDE, Message
int[] to = new int[] {R.id.message_row, R.id.sender_row, R.id.latitude_row, R.id.longitude_row};
cursorAdapter = new SimpleCursorAdapter(getActivity(), R.layout.message_row, null, from, to, 0);
```

Figure 7.1

```
@Override
public void onViewCreated(View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    final Client client = getArguments().getParcelable(TAG);
    MESSAGES_FRAGMENT_LOADER_ID = (int)client.id + 10;

    nameView = (TextView)view.findViewById(R.id.messages_client_name);
    nameView.setText(client.name);
    addressView = (TextView)view.findViewById(R.id.messages_client_address);
    addressView.setText(client.address);
```

Figure 7.2

8. To enable the unregister operation, define a new class called Unregister which extends the class Request as Figure 8.1. Implement the unregister operation as register operation except in the RestMethod class, the HttpURLConnection's request method should be "DELETE" as Figure 8.2 shows. When the unregister operation is successful, a toast message will notify the user.

```java
public class Unregister extends Request {
    public static final String TAG = Unregister.class.getCanonicalName();
    public String host;
    public int port;

    public static final Creator<Unregister> CREATOR = new Creator<Unregister>() {
        @Override
        public Unregister createFromParcel(Parcel source) { return new Unregister(source); }

        @Override
        public Unregister[] newArray(int size) { return new Unregister[0]; }
    };

    public Unregister(Parcel parcel) {
        this.host = parcel.readString();
        this.port = parcel.readInt();
        ParcelUuid parcelUuid = parcel.readParcelable(ParcelUuid.class.getClassLoader());
        this.registrationID = parcelUuid.getUuid();
        this.clientID = parcel.readLong();
        this.latitude = parcel.readDouble();
        this.longitude = parcel.readDouble();
    }

    public Unregister(String host, int port, Client client) {
        this.host = host;
        this.port = port;
        this.registrationID = client.uuid;
        this.clientID = client.id;
        this.latitude = client.latitude;
        this.longitude = client.longitude;
    }

    @Override
    public Uri getRequestUri() {
        String requestString = "http://" + host + ":" + String.valueOf(port) + "/chat/" +
                String.valueOf(clientID) +
                "?";
        try {
            requestString += "regid=" + URLEncoder.encode(registrationID.toString(), encoding);
        } catch (UnsupportedEncodingException e) {
            Log.e(TAG, e.getMessage());
        }
        Log.v(TAG, "Unregister Request URI: " + requestString);
        return Uri.parse(requestString);
    }
```

Figure 8.1

```
public boolean perform(Unregister request) {
    URL url = request.getRequestUrl();
    if (isOnline()) {
        try {
            // prepare request
            connection = (HttpURLConnection)url.openConnection();
            connection.setRequestProperty("USER_AGENT", TAG);
            connection.setRequestMethod("DELETE");
            connection.setUseCaches(false);
            connection.setRequestProperty("CONNECTION", "Keep-Alive");
            connection.setConnectTimeout(15000);
            connection.setReadTimeout(10000);

            Map<String, String> headers = request.getRequestHeaders();
            for(Map.Entry<String, String> header : headers.entrySet()) {
                connection.addRequestProperty(header.getKey(), header.getValue());
            }

            // execute request
            outputRequestEntity(request);
            connection.setDoInput(true);
            connection.connect();
            throwErrors(connection);
            if (connection.getResponseCode() < 200 || connection.getResponseCode() >= 300) {
                connection.disconnect();
                return false;
            } else {
                connection.disconnect();
                return true;
            }
        } catch (IOException e) {
            Log.e(TAG, e.getMessage());
        }
    }
    return false;
}
```

Figure 8.2

9. In the ChatAppActivity, override the onPrepareOptionsMenu to check if the register or unregister button should be visible as Figure 9.1. When the unregister button is clicked, it will implement the UnregisterClient method to unregister the user from the server as Figure 9.2.

```
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    if (client == null) {
        menu.findItem(R.id.unregister).setVisible(false);
        menu.findItem(R.id.action_settings).setVisible(true);
    } else {
        menu.findItem(R.id.unregister).setVisible(true);
        menu.findItem(R.id.action_settings).setVisible(false);
    }
    return true;
}
```

Figure 9.1

```
private void UnregisterClient() {
    if (client != null) {
        Unregister unregister = new Unregister(host, port, client);
        receiver = (IReceiver) (resultCode, resultData) -> {
                if (resultCode == RequestService.RESULT_UNREGISTER_OK){
                    SharedPreferences.Editor editor = getSharedPreferences(SettingDialogFragment.MY_SHARED_PREF, MODE_
                    editor.putString(SettingDialogFragment.PREF_USERNAME, "");
                    editor.putLong(SettingDialogFragment.PREF_IDENTIFIER, -1);
                    editor.apply();
                    reSetInfo();
                    MessageManager manager = new MessageManager(ChatAppActivity.this, (cursor) -> {
                            return new Message(cursor);
                    }, CHAT_APP_LOADER_ID);
                    manager.DeleteAllAsync();
                    Toast.makeText(ChatAppActivity.this, "Unregister successfully", Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(ChatAppActivity.this, "Unregister failed", Toast.LENGTH_SHORT).show();
                }
        };
        wrapper = new AckReceiverWrapper(new Handler());
        wrapper.setReceiver(receiver);
        serviceHelper.UnregisterUser(unregister, wrapper);
    }
}
```

Figure 9.2

10. Run and test the App as the video shows.