APPENDIX C: JSON AND DATABASE SCHEMA

I. SURVEY/DEMOGRAPHICS CENTRAL DATABASE TABLES

The overall architecture is designed to simplify the codebase by using JSON primarily as a transport vehicle and limiting the number of database fields to those that need to be known by the database in question. For example, the AWS source database for downloading questions only needs to know QUESTIONGUID (for a hash key) and the JSON_STR containing the meat of the question. Telling it ordinal-position simplifies other areas of the Android code and so that was included. Otherwise, the content remains unparsed until downloaded by the Android app.

On device, the database is again limited to guid, ordinal-position, and JSONSTR. The additional fields are flags for internal tracking use. Parsed JSON supplies fields for the generation of the question on device and for the uploaded response—

HTTP://WWW.JSONEDITORONLINE.ORG/
HTTPS://WWW.GUIDGENERATOR.COM/ONLINE-GUID-GENERATOR.ASPX

AWS QUESTION TABLE:

- QUESTIONGUID_STR : STRING, PRIMARY HASH KEY
- ORDINALPOSITION_INT : INT, RANGE KEY
- JSON_STR : STRING

JSON STR JSON SCHEMA: QUESTION

{  
    SURVEYNAME_STR : STRING
    --- SURVEYGUID_STR : STRING
    --- ORDINALPOSITION_INT : INT
    --- QUESTIONGUID_STR : STRING
    --- QUESTIONPROMPT_STR : STRING
    --- QUESTIONTYPE_STR : STRING
    --- RESPONSES_ARR : ARRAY
    ---
        {  
            RESPONSEID_INT : INTEGER
            --- RESPONSE_STR : STRING
        ---
        ---
        ---
        }
}

\--- PUBLIC-STATIC FINAL STRING TYPE_BUTTONS = "buttons";
\--- PUBLIC-STATIC FINAL STRING TYPE_TEXT = "text";
\--- PUBLIC-STATIC FINAL STRING TYPE_CHECKBOXES = "checkboxes";
\--- PUBLIC-STATIC FINAL STRING TYPE_SLIDER = "slider";
OPTIONALLY

--- CONDITIONAL_UPON_QUESTIONGUID_STR : STRING // QUESTIONGUID TO CHECK*
--- CONDITIONAL_UPON_RESPONDID_ARR : ARRAY // RESPONDIDS TO CHECK*

----------{
    
    CONDITIONAL_UPON_RESPONDID_INT : INTEGER

    }
----------

CONDITIONAL BY COUNTRY_STR : STRING // USE ISO-3166 ALPHA-2 CODES

CONDITIONAL_UPON_DATEMSID_INT : INTEGER
    // DATE (IN UTC UNIX EPOCH MS) AFTER WHICH TO ADMINISTER THIS QUESTION

DELAYBYSAYS_INT : INTEGER
    // WAIT THIS MANY DAYS AFTER THE QUESTION IS FIRST DOWNLOADED TO ASK THIS QUESTION

ONGOINGQUESTION_ARR : ARRAY // ARRAY OF DAY OF WEEK + TIME AS FOLLOWS

----------{
    
    NOTIFICATIONTIME_STR : STRING

    }
----------

// NOTIFICATIONTIME FORMATTED AS FOLLOWS: EEEHHmm

// EEE = THREE LETTER DAY OF WEEK (Mon, Tue, Wed, Thu, Fri, Sat, Sun, Dly)

// DLY = DAILY

// HH = Hours 00-23

// MM = Minutes 00-59

// EXAMPLES: Tue0900, Thu1400, DLY1200

DELETEQUESTION_STR : STRING // QUESTIONGUID OF ONGOING QUESTION TO
    // DELETE FROM LOCAL SQLITE DB

}

LOCAL DB ON ANDROID

Table Questions

QUESTIONGUID_STR

JSON_STR

ORDINALPOSITION_INT

FINAL_RESPONDID_INT

FINAL_RESPONSE_STR

ANSWERED_BOOL

Table Responses

ID

JSON

UPDATED
II. RESPONSES: GENERIC RESPONSE SCHEMA

The generic schema serves as the basic information passed with all types of uploaded data. The additional overhead is minimal and the presence of this information in each of uploaded packet simplifies future analysis against unnecessary complexity in terms of secondary lookups. Most NoSQL solutions do not allow joins.

```
{
    USERGUID_STR : STRING //
    LOCALTIME_MS_INT : INTEGER // PRIMARY RANGE INDEX
    LOCALTIME_HRS_MILITARY_INT : INTEGER
    LOCALTIME_DAYOFWEEK_STR : STRING
    LOCALTIMEZONE_STR : STRING
    COUNTRY_TM_STR : STRING // FROM TELEPHONE MANAGER
    LO_LANG_STR : STRING // LOCAL LANG
    REGION_IPAPI_STR : STRING // WWW.IP-API.COM/JSON
    REGIONNAME_IPAPI_STR : STRING
    COUNTRY_IPAPI_STR : STRING // FROM IPAPI
    REGION_GC_STR : STRING // FROM GEOCODING
    COUNTRY_GC_STR : STRING // FROM GEOCODING
    LATITUDE_GC_FLOAT : FLOAT // EASILY EXCLUDED USING BOOLEAN
    LONGITUDE_GC_FLOAT : FLOAT // EASILY EXCLUDED USING BOOLEAN
    LATLONACCURARY_GC_FLOAT : FLOAT // EASILY EXCLUDED USING BOOLEAN
    ENTRYTYPE_STR : STRING // PRIMARY HASH INDEX
    ...
}
```

2. GUID unique to the installation, which allows a unique user’s behavior over time to be analyzed while making it practically impossible to identify that user.
III. Responses: Specific added fields to generic document schema

Questions

ENTRYTYPE_STR : "SURVEY",
SURVEYGID_STR : STRING,
QUESTIONGUID_STR : STRING,
QUESTIONPROMPT_STR : STRING,
RESPONSE_STR : STRING,
RESPONSEID_STR : STRING // QUESTIONGUID_STR & " " INTEGER.to.String(RESPONSEID_INT)
RESPONSES_ARR : ARRAY [if type is multiple response eg checkbox)
[{
    RESPONSEID_STR : STRING
    // QUESTIONGUID_STR & " " INTEGER.to.String(RESPONSEID_INT)
    RESPONSE_STR : STRING
    }
    RESPONSEID_STR : STRING
    // QUESTIONGUID_STR & " " INTEGER.to.String(RESPONSEID_INT)
    RESPONSE_STR : STRING
    ]
]

Consent/Consent-change

ENTRYTYPE_STR : "CONSENTCODE_INT/CONSENTCHANGE_INT"
CONSENTCODE_INT : INTEGER
CONSENTCHANGE_INT : INTEGER
1 - DO NOT CONSENT
2 - CONSENT
3 - EXIT STUDY
4 - RE-ENTER STUDY

On-start

ENTRYTYPE_STR : "ONSTART"
AGE_YRS_FRA : FRACTION
WEIGHT_KG_FRA : FRACTION

Total-time-using-the-app

ENTRYTYPE_STR : "TOTALTIMEOFUSE",
TIMEINAPP_MS_INT : INTEGER,

IV. Modifying for new responses
In order to add response fields to the JSON document in the AWS Responses table, follow these steps:

1. Add the new field to the database schema in AWSConstants:
   ```java
   public static final String DBSCHEMA_MYNEWENTRYTYPE = "MYENTRYTYPE";
   public static final String DBSCHEMA_MYNEWFIELD = "MY_NEW_FIELD_STR";
   ```

2. Make the call to ResponseSenderAsyncTaskService in your code:
   ```java
   JSONObject JC = new JSONObject();
   String MYDATA = "MYDATA";
   try {
       JC.put(SA_AWSCONSTANTS.DBSCHEMA_ENTRYTYPE,  
               SA_AWSCONSTANTS.DBSCHEMA_MYNEWENTRYTYPE);
       JC.put(SA_AWSCONSTANTS.DBSCHEMA_MYNEWFIELD,  
               MYDATA);
   } catch (JSONException e) {} 
   Intent c = new Intent(getBaseContext(), SA_ResponseSenderAsyncTaskService.class);
   c.putExtra(SA_AWSCONSTANTS.INTENT_EXTRA_JSONSTRING, JC.toJSONString());
   startService(c);
   ```

3. Add the variable for the data to Response.java. Create getter and setter including DynamoDB tag:
   ```java
   NOTE: If the data is numeric or boolean, use the nullable "Boolean", "Double" or "Integer" as opposed to the primitive "Boolean", "Double" or "Integer".
   private String MYDATA;
   @DynamoDBAttribute  
   (AttributeName = SA_AWSCONSTANTS.DBSCHEMA_MYNEWFIELD)  
   public String getMyDATA() {  
       return MYDATA;
   }
   
   public void setMyDATA(String tMYDATA) {  
       this.MYDATA = tMYDATA;
   }
   ```

4. Add the following conditional statement to the constructor Response(JSONObject j):
   ```java
   if (j.has(SA_AWSCONSTANTS.DBSCHEMA_MYNEWFIELD)) {
       USERGUID_STR = j.getString(SA_AWSCONSTANTS.DBSCHEMA_MYNEWFIELD);
   }
   ```