**DEVELOPERS MANUAL**

Expanded National Nutrition Survey (ENNS) 2018: Anthropometry

**Food and Nutrition Research Institute Department of Science and Technology**

**TABLE OF CONTENTS**

1. GENERAL INFORMATION
   1. System Overview
   2. Definition of Technical Term
2. SYSTEM SUMMARY
   1. System Configuration
   2. Public Directory
3. SYSTEM ARCHITECTURE
   1. Database Connection Constants
   2. Classes, Methods and Functions
      * 1. Server Connection
        2. Login Validation
        3. Auto complete
        4. Household list
        5. Members list
        6. Summary of households and members
        7. Back-up data
        8. Consolidate data
        9. Transmit data
        10. Form questions
        11. Form answers
        12. Form skipping

1. SYSTEM DATABASE
   * + 1. Membership form
       2. Listing of households
       3. Household forms
       4. Individual forms
       5. Answers
       6. Questions
       7. Skipping
       8. Login
       9. Localsurveyareas
       10. Login

**1.0 GENERAL INFORMATION**

**1. GENERAL INFORMATION**

**1.1 System Overview**

This program is part of Electronic Data Collection System (eDCS) implemented by FNRI to reduce the use of paper-based forms and to save time for processing of data by transmitting through internet.

**1.2 Definition of Technical Term**

This section defines the common terms used during the development of the system.

|  |  |
| --- | --- |
| Autoload | Is the capability of loading libraries, helpers and models automatically every time the system will run. |
| Class | Is use in object-oriented programming to describe one or more objects. |
| Database | Is a storage and large collection of data to ease and speed of search and retrieval. |
| Function | Is a block of statement that can be use over and over again in a program |
| Method | Is a specific type of function and it is part of a class. |
| Object | Is a combination of variables, functions or data structures. |
| Parameters | Is a variable in a function that has multiple values depending on conditions. |
| Session | Is a way to store data for individual user that can use in every page requests. |
| Variable | Is a value that can change depending on information passed to the program. |

**2.0 SYSTEM SUMMARY**

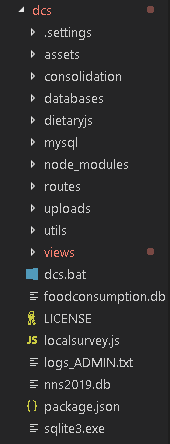
**2. SYSTEM SUMMARY**

**2.1 User Access Levels**

The administrator can view all the columns in all tables. While encoder have limited views only and can encode only in a particular columns in tables.

**2.2 Public Directory**

This directory contains all file used in the program such as fonts, images, layouts, stylesheets, and scripts. (Location : C:\dcs)



**3.0 SYSTEM ARCHITECTURE**

**3. SYSTEM ARCHITECTURE**

**3.1 Database Connection Constants**

The database configuration can be found in nns2019.db in dcs folder. Database connection string are necessary to declare as constant to access them globally. Database connection on SQLite should be declared like this (the developer shows a random file in this method; form 1.4)

var sqlite3 = require('sqlite3').verbose();

db2 = new sqlite3.Database('nns2019.db');

exports.get = function(req, res){

    var eacode = req.params.id;

    var eacode2 = eacode.substring(0, 12);

    var hcn = req.params.id;

    var hcn2 = hcn.substring(12, 16);

    var shsn = req.params.id;

    var shsn2 = shsn.substring(16, 20);

db2.all("SELECT \* FROM f14 WHERE eacode = ? and hcn= ? and shsn= ?",[eacode2, hcn2, shsn2], function(err,rows){

db2.all("SELECT \* FROM questions WHERE qkey LIKE '%f14\_%' ORDER BY qkey ASC", function(err,rows3){

        db2.all("SELECT \* FROM answers WHERE qkey LIKE '%f14\_%' order by variable\_name ASC", function(err,rows4){

                res.render('form14.ejs',{array:rows,array3:rows3,array4:rows4,layout:false,session:req.session});

        });

    });

});

};

**3.2 Functions**

This section list down and describes the different classes including their methods and functions used during the development of the system.

**3.2.1 Server connection**

This file facilitates the communication between the server-side to client-side server and performs the query requested from the client-side.

***server function (localsurvey.js):***

var express = require('express');

var routes = require('./routes');

var login = require('./databases/login');

var summarylist = require('./databases/summarylist');

var exports = require('./databases/exports');

var transmission = require('./databases/transmission');

var consolidation = require('./databases/consolidation');

var app = express();

app.set('port', process.env.PORT || 3000);

app.set(process.env.NODE\_TLS\_REJECT\_UNAUTHORIZED = "0");

app.set('views', \_\_dirname + '/views');

app.set('view engine', 'ejs');

app.use(express.favicon());

app.use(express.logger('dev'));

app.use(bodyParser.json());

app.use(express.cookieParser());

app.use(bodyParser.urlencoded({ extended: true }));

app.use(express.json());

app.use(bodyParser.json());

app.use(express.methodOverride());

app.use(express.static(path.join(\_\_dirname, 'assets')));

app.use(express.static(path.join(\_\_dirname)));

app.post('/upload\_cons', upload.single('userfile'), consolidation.extract);

app.post('/upload', upload.single('csvdata\_cons'), consolidation.upload);

app.post('/upload\_trans', upload.single('userfile'), transmission.extract);

app.post('/upload\_transD', upload.single('userfile'), transmission.extractD);

app.post('/upload\_email', upload.single('userfile'), transmission.email);

app.post('/trans', upload.single('csvdata\_trans'), transmission.upload);

app.post('/transD', upload.single('csvdata\_transD'), transmission.uploadD);

app.get('/checktrans/:id', transmission.get);

app.get('/checktransD/:id', transmission.getD);

app.post('/membersTL', transmission.membersTL);

app.post('/f11T', transmission.f11T);

app.post('/f11A', transmission.f11A);

app.post('/login', login.post);

http.createServer(app).listen(app.get('port'), function(req, res){

    console.log('Express server listening on port ' + app.get('port'));

});

**3.2.2 Login Validation**

This method will verify if the username exists on database. If row exists, the username and status will declare on session and it redirect to home page else it will redirect to login page.

***Login function:***

var fs = require('fs');

var path = require('path');

var sqlite3 = require('sqlite3').verbose();

db2 = new sqlite3.Database('nns2019.db');

var http = require('http');

var session = require('express-session');

var dateLog = new Date().toLocaleString();

exports.post = function (req, res){

    if(req.body.username != '' && req.body.pword != '')

    {

         var username = req.body.username;

         var password = req.body.pword;

            db2.all("SELECT \* FROM login WHERE username = ? and type = ?", [username, password], function(err, rows) {

            if(rows.length > 0 ) {

            if(err){ res.redirect('/'); }

                fs.appendFile('logs\_'+req.session.username+'.txt', '['+dateLog+']['+req.body.username+'] Logged in as user '+req.body.username+' \n');

                                    req.session.username = req.body.username;

                                    res.redirect('/legone/survey/surveyform/');

            }

            else {

                fs.appendFile('logs.txt', '['+dateLog+'] Failed login \n');

                res.redirect ( '/');

            }

            });

    }

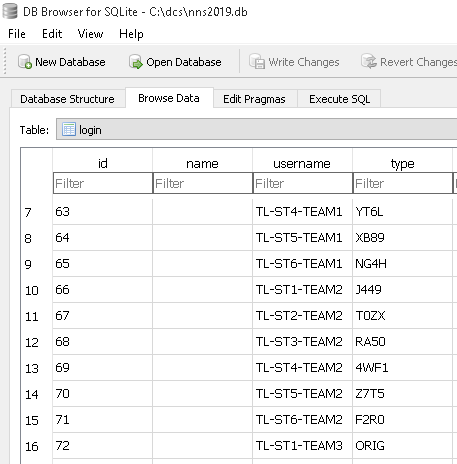
    else{

        res.redirect('/');

    }

};

***Login database:***



**3.2.3 Autocomplete**

This method call the method autocomplete in the model **listareas.js** and send to client-side the returned data.

var fs = require('fs');

var path = require('path');

var sqlite3 = require('sqlite3').verbose();

db2 = new sqlite3.Database('nns2019.db');

var http = require('http');

exports.post = function(req, res){

    var obj = [];

    var input = req.body.areaname;

    if (input == "") {

        res.send("nodata");

    }

    else

    {

    db2.all("SELECT \* FROM localsurveyareas WHERE areaname LIKE '%"+ input +"%' or eacode like '%"+ input +"%' LIMIT 20", function(err,rows){

    for (var i in rows)

        {

            obj.push(rows[i]);

        }

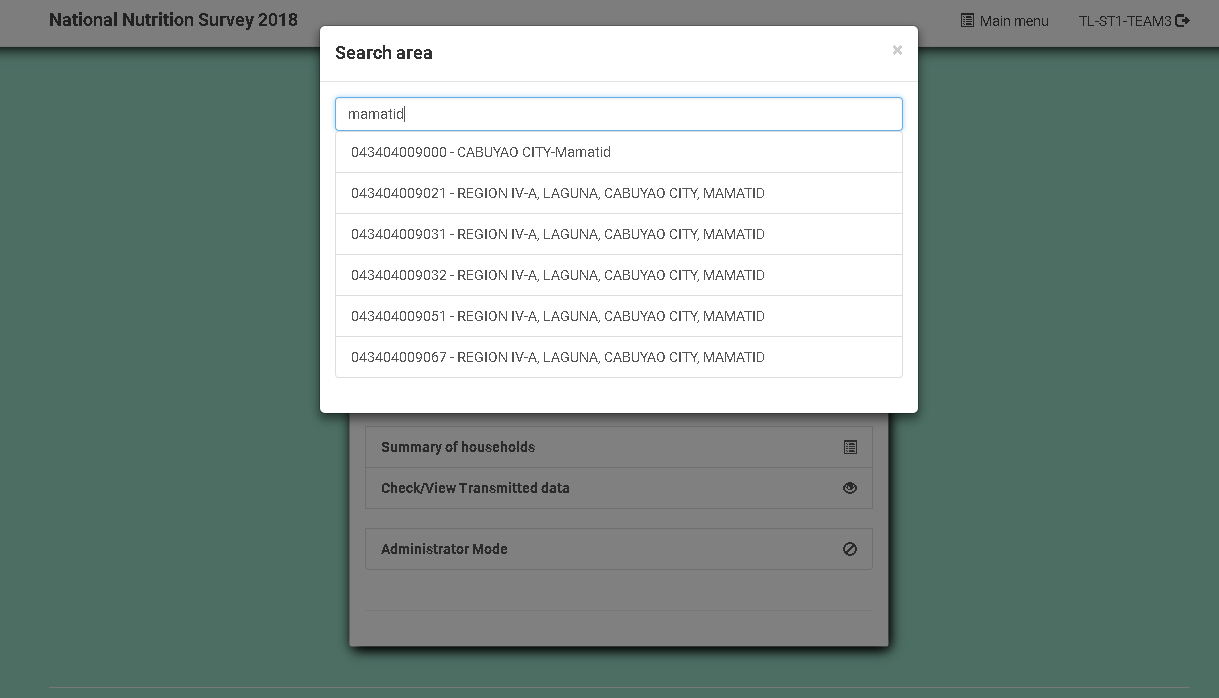
         res.send(obj);

        });

    }

};

***Sample output:***



**3.2.4 Households List**

This method selects and sends the returned data of all household head by area code or by area name to client-side.

***Household function (household.js):***

var fs = require('fs');

var path = require('path');

var sqlite3 = require('sqlite3').verbose();

db2 = new sqlite3.Database('nns2019.db');

var http = require('http');

var dateLog = new Date().toLocaleString();

exports.get = function(req, res){

    var id = req.params.id;

    fs.appendFile('logs\_'+req.session.username+'.txt', '['+dateLog+'] Household Listings : '+req.params.id+' \n');

    db2.all("SELECT \* FROM localarea\_listings WHERE eacode like '%"+[id.substring(0,9)]+"%' and eacode like '%P%' order by eacode", function(err,rows){

        db2.all("SELECT \* FROM f11 WHERE eacode like '%"+[id.substring(0,9)]+"%' and eacode like '%P%'", function(err,rows2){

            db2.all("SELECT \* FROM login WHERE eacode like '%"+[id.substring(0,9)]+"%' and eacode like '%P%'", function(err,rows3){

                db2.all("SELECT \* FROM localsurveyareas WHERE eacode like '%"+[id.substring(0,9)]+"%' and eacode like '%P%'", function(err,rows4){

                    res.render('households.ejs',{array:rows,array2:rows2,array3:rows3,array4:rows4,layout:false,session:req.session});

                });

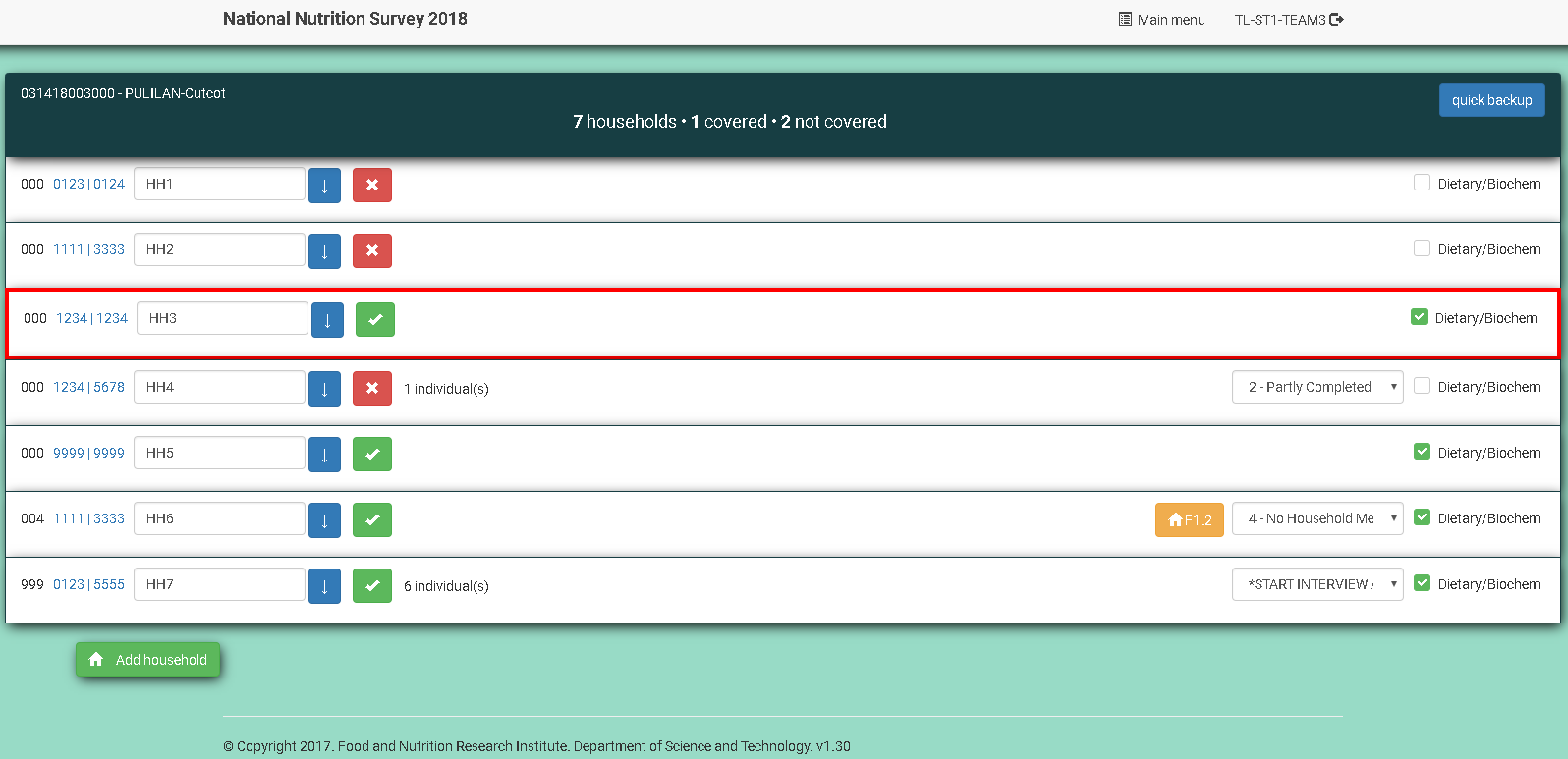
            });

        });

    });

};

***Sample output (households.ejs):***



Information seen per household (from left to right):

1. EA – last 3 characters of eacode
2. HCN/SHSN – Will go to membership (if clicked)
3. Household head – Editable + Auto save
4. Call-back function – the user will enter 3 call-backs (if applicable)
5. Other information needed – latitude, longitude, address, replacement household (if applicable)
6. Displays number of members of household
7. F.12 button – it will appear if the household has a status of not covered + Auto save
8. Final IS – It will appear if the household has completed call-back
9. Dietary/Biochem checkbox – checked or unchecked depending on the data given by Statisticians

Additional Features:

1. Displays total households tally with data
2. Quick backup
3. Add household (if needed)
   * 1. **Members list**

This method selects and sends the returned data of all members by eacode, hcn and shsn by clicking the HCN/SHSN on households page.

***Members function (form11.js):***

exports.get = function(req, res){

    var eacode = req.params.id;

    var eacode2 = eacode.substring(0, 12);

    var hcn = req.params.id;

    var hcn2 = hcn.substring(12, 16);

    var shsn = req.params.id;

    var shsn2 = shsn.substring(16, 20);

    fs.appendFile('logs\_'+req.session.username+'.txt', '['+dateLog+'] Membership : '+eacode2+' | '+hcn2+' | '+shsn2+' \n');

db2.all("SELECT \* FROM localarea\_listings WHERE eacode = ? and hcn= ? and shsn= ?",[eacode2, hcn2, shsn2], function(err,rows){

    db2.all("SELECT \* FROM f11 WHERE eacode = ? and hcn= ? and shsn= ?",[eacode2, hcn2, shsn2], function(err,rows2){

        db2.all("SELECT \* FROM survey\_forms WHERE table\_name='f11'", function(err,rows3){

            res.render('form11.ejs',{array:rows,array2:rows2,array3:rows3,layout:false,session:req.session});

                    // res.render('form11.ejs',{array:rows,array2:rows2,array3:rows3});

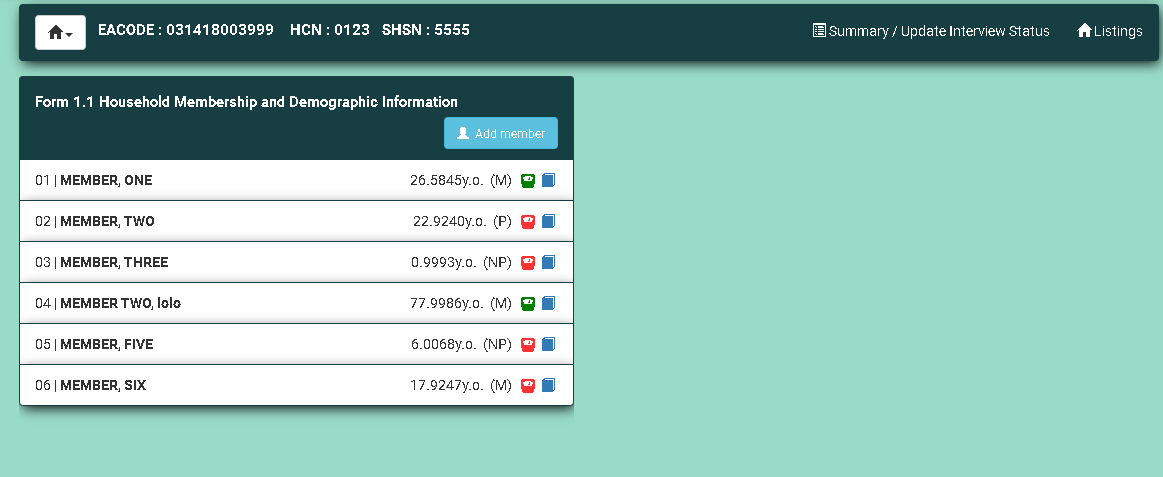
        });

    });

});

};

***Sample output (form11.ejs):***



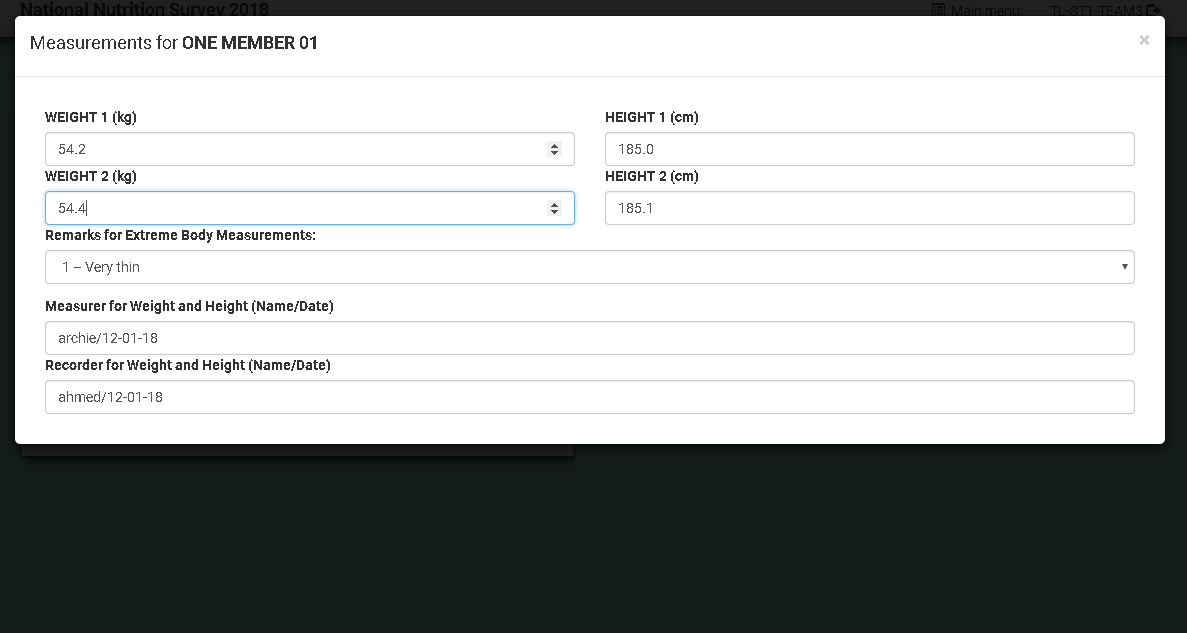
Information seen per Member (from left to right):

1. Member code
2. Member name (last name, first name)
3. Exact age
4. Physiological code
5. Anthropometric Measurements and Blood Pressure **(Picture 1)**
6. Individual Forms **(Picture 2)**

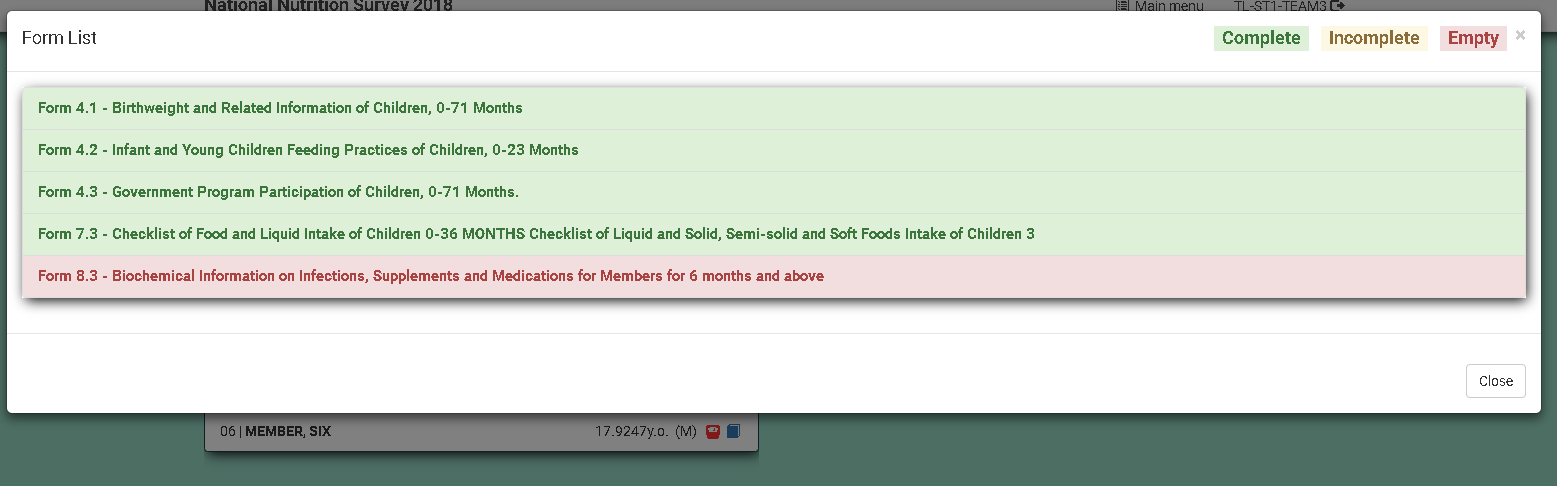
Additional Features:

1. Household forms can be filled-up in the house icon drop down **(Picture 3)**
2. Summary of members link
3. Households link

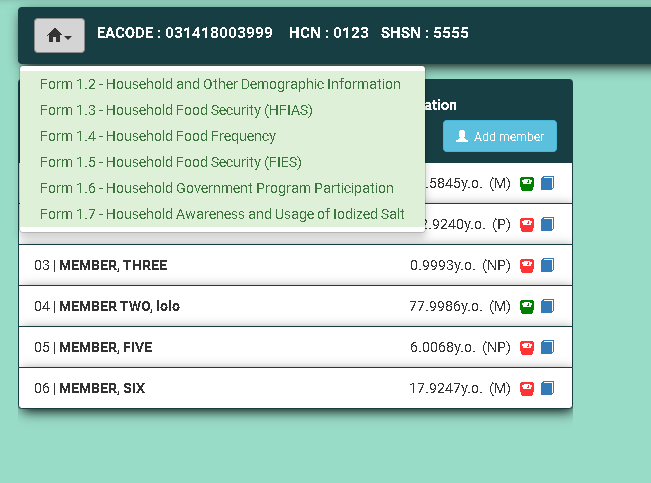
**Picture 1**



**Picture 2**



**Picture 3**



**3.2.6 Summary of Household and Members**

This method selects and sends the returned data of all members by eacode, hcn and shsn by clicking Summary link on membership page.

***Members function (summary\_indiv.js):***

exports.get = function(req, res){

    var eacode = req.params.id;

    var eacode2 = eacode.substring(0, 12);

    var hcn = req.params.id;

    var hcn2 = hcn.substring(12, 16);

    var shsn = req.params.id;

    var shsn2 = shsn.substring(16, 20);

    var MEMBER\_CODE = req.params.id;

    var MEMBER\_CODE2 = MEMBER\_CODE.substring(20,22);

db2.all("SELECT \* FROM localarea\_listings WHERE eacode = ? and hcn= ? and shsn= ?",[eacode2, hcn2, shsn2], function(err,rows){

        db2.all("SELECT \* FROM f11 WHERE eacode = ? and hcn= ? and shsn= ? and MEMBER\_CODE = ?",[eacode2, hcn2, shsn2, MEMBER\_CODE2], function(err,rows2){

            db2.all("SELECT \* FROM localsurveyareas WHERE eacode = ?",[eacode2], function(err,rows3){

            res.render('summary\_indiv.ejs',{array:rows,array2:rows2,array3:rows3,layout:false,session:req.session});

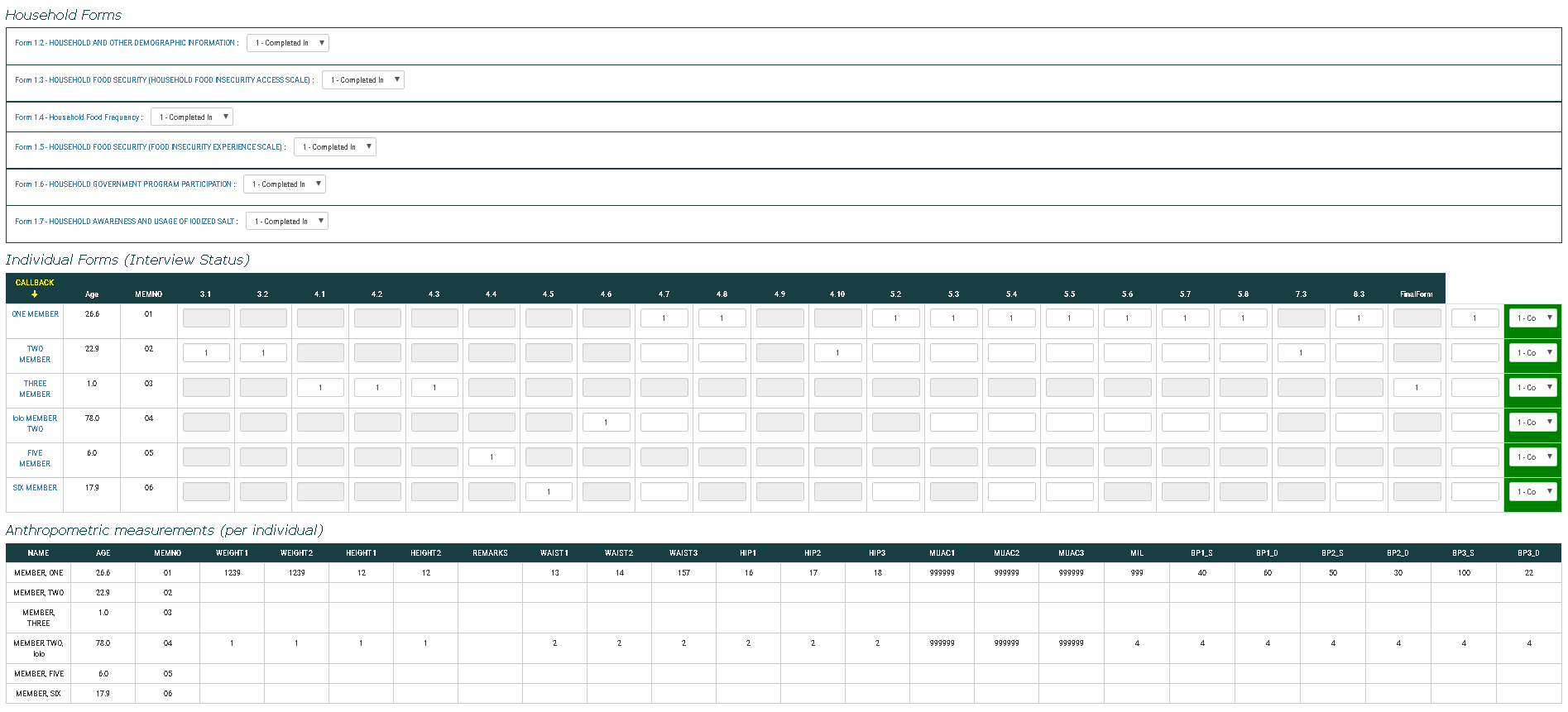
        });

     });

});

};

***Sample output (summary\_indiv.ejs):***



**3.2.7 Back-up data**

This method selects and sends the returned data of all households with data clicking Export and backup data button on main menu page. The user can back up data by clicking what component he/she wants to download

***Members function (exports.js):***

exports.get = function (req, res) {

    db2.all("select distinct localarea\_listings.eacode, localsurveyareas.areaname from localarea\_listings INNER JOIN localsurveyareas ON localarea\_listings.eacode = localsurveyareas.eacode WHERE INTERVIEW\_STATUS1 > 0 ORDER BY localarea\_listings.eacode", function (err, rows) {

        if (err)

            console.log(err);

        res.render('export.ejs', {

            array: rows,

            layout: false,

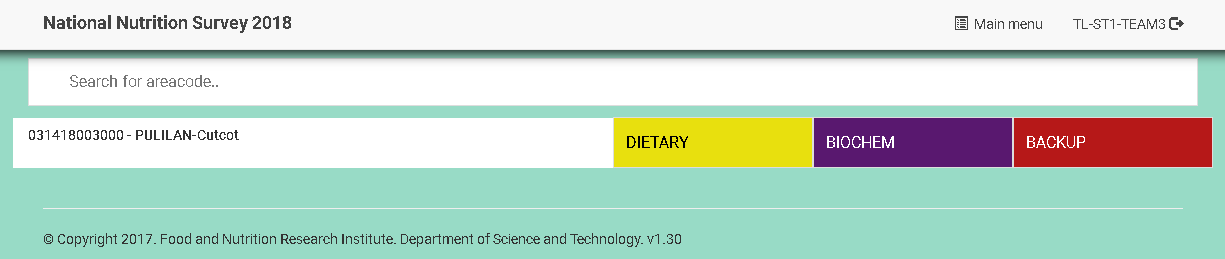
            session: req.session

        });

    });

};

***Sample output (export.ejs):***



Information seen per household (from left to right):

1. Eacode and Area name information
2. Dietary button – will backup form 1.1 data for dietary system encoding
3. Biochem button – will backup form 1.1 and localarea\_listings data for biochem system encoding
4. Backup button – will back up ALL data for consolidating and transmitting data

**3.2.8 Consolidate data**

This method sends the data from the zip file uploaded by the user to his/her team leader’s database. These are the steps in the function:

***consolidation function (consolidation.js):***

1. Extracting the uploaded zip file

exports.extract = function (req, res, next) {

  csvFilePath=req.file.path;

  fs.createReadStream(csvFilePath).pipe(unzip.Extract({ path: 'C:/dcs/consolidation' }));

};

1. Reads the extracted files

exports.upload = function (req, res, next) {

var array\_name = ['f11','f12','f13','f14','f15','f16','f17','f21','f31','f32','f41','f42','f43','f44','f45','f46','f47','f48','f49','f410','f411','f52','f53','f54','f55','f56','f57','f58','f59','f73','f82','localarea\_listings'];

var array\_length = array\_name.length;

fs.appendFile('logs\_'+req.session.username+'.txt', '['+dateLog+'] Consolidating '+req.file.originalname+' \n');

for(var i = 0 ; i< array\_length ; i++){

var name = array\_name[i]+"\_"+req.file.originalname.substring(0,req.file.originalname.length-8)+".csv";

csvFilePath='C:/dcs/consolidation/'+array\_name[i]+"\_"+req.file.originalname.substring(0,req.file.originalname.length-8)+".csv";

var form\_name = array\_name[i];

var username = req.session.username;

csv\_upload(name,form\_name,username);

}

res.write('<style>.one { opacity: 0; -webkit-animation: dot 1.3s infinite; -webkit-animation-delay: 0.0s; animation: dot 1.3s infinite; animation-delay: 0.0s; }.two { opacity: 0; -webkit-animation: dot 1.3s infinite; -webkit-animation-delay: 0.2s; animation: dot 1.3s infinite; animation-delay: 0.2s; }.three { opacity: 0; -webkit-animation: dot 1.3s infinite; -webkit-animation-delay: 0.3s; animation: dot 1.3s infinite; animation-delay: 0.3s; }@-webkit-keyframes dot { 0% { opacity: 0; } 50% { opacity: 0; } 100% { opacity: 1; } }@keyframes dot { 0% { opacity: 0; } 50% { opacity: 0; } 100% { opacity: 1; } }</style><body>');

  res.write('<div style="height: 200px;width: 400px;position: fixed;top: 50%;left: 50%;margin-top: -100px;margin-left: -200px;"><font size="7">Consolidating data Please Wait<span class="one">.</span><span class="two">.</span><span class="three">.</span></font> </div></body><script type="text/javascript"> window.location.href="/legone/survey/surveyform/";</script>');

res.end()

//...

//res.redirect('/legone/survey/surveyform/');

};

1. Perform passed function

function csv\_upload(name,form\_name,username){

csv()

.fromFile(csvFilePath)

.on('json',(jsonObj)=>{

JSON2Sqlite(jsonObj,name,form\_name,username);

})

.on('done',(error)=>{

console.log('Upload Success');

})

};

1. Insert data per form into the team leader’s database

function JSON2Sqlite(json,name,form\_name,username){

var col = [];

var values =[];

var sql ="";

var rn = 0;

var arr = name.split('\_');

for (var k in json){

if(k=='id'){

// console.log(sql);

if(col.length>0){

db2.all(sql ,function(err,rows){

if(err) //console.log(err);

rn++;

});

col = [];

values = [];

}

}

else{

col.push(k);

values.push(json[k]);

}

}

if(col.length>0){

sql = "INSERT OR IGNORE INTO "+form\_name+"(" + col.join(",") + ") VALUES('" + values.join("','") + "');";

db2.all(sql ,function(err,rows){

if(err) { console.log("Error number : " + err);

fs.appendFile('logs\_'+username+'.txt', '['+dateLog+'] Error on consolidation on '+form\_name+' : '+err+' \n');

}

rn++;

});

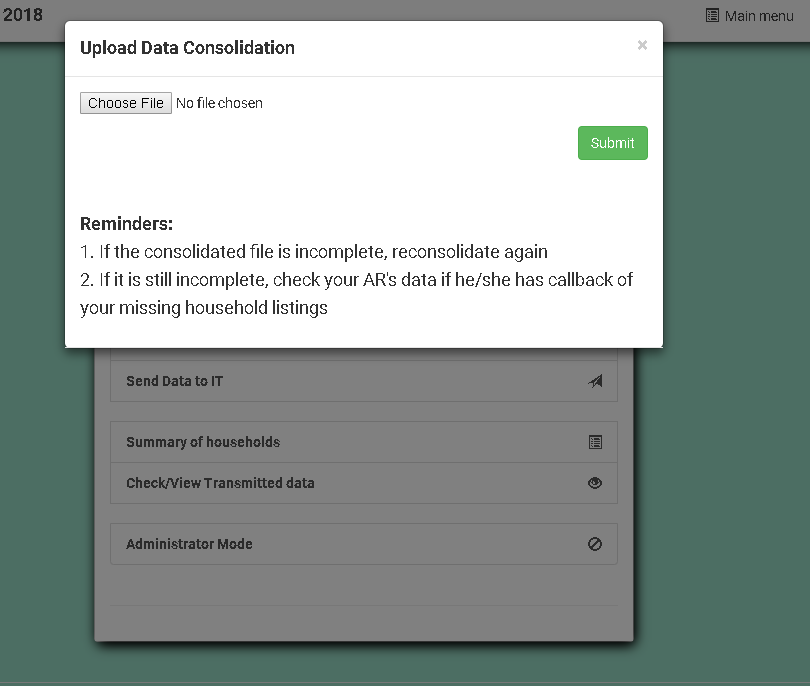
col = [];

values = [];

}

}

***Sample output (localsurvey.ejs):***



**3.2.9 Transmit data**

This method sends the data from the zip file uploaded by the users to the FNRI server. These are the steps in the function:

***transmission function (transmission.js):***

1. Connect to the FNRI Server

var mysql = require('mysql');

var connection = mysql.createConnection({

host : '122.53.86.117',

port   : '3674',

user : 'johncarlo',

password : 'nsis',

database : 'nns2018db'

});

1. Extracting the uploaded zip file

exports.extract = function (req, res, next) {

    csvFilePath=req.file.path;

    var fx = fs.createReadStream(csvFilePath).pipe(unzip.Extract({ path: 'C:/dcs/consolidation' }));

    fx.on('error', function(err) {

        fs.appendFile('logs\_'+req.session.username+'.txt', '['+dateLog+'] Error extracting file for transmission '+req.file.originalname+' : '+err+' \n');

        res.send('error');

    });

};

1. Sends email to NAMD-NSIS account

exports.email = function (req, res, next) {

    csvFilePath=req.file.path;

    'use strict';

const nodemailer = require('nodemailer');

// Generate test SMTP service account from ethereal.email

// Only needed if you don't have a real mail account for testing

nodemailer.createTestAccount((err, account) => {

// create reusable transporter object using the default SMTP transport

let transporter = nodemailer.createTransport({

host: 'smtp.gmail.com',

port: 587,

secure: false, // true for 465, false for other ports

auth: {

user: 'enns.fnridost@gmail.com', // generated ethereal user

pass: 'namdnsis3' // generated ethereal password

}

});

// setup email data with unicode symbols

let mailOptions = {

from: 'enns.fnridost@gmail.com', // sender address

to: 'fnri.namd.nsis.trans@gmail.com', // list of receivers password:namdnsis2019

subject: 'FROM : '+ req.session.username, // Subject line

text: 'Transmitted data', // plain text body

html: 'Transmitted data sent by '+req.session.username+'. Please download attached file (in zip)', // html body

        attachments: [{path:"C:/Users/ENCODER/Downloads/"+req.file.originalname, type:"application/zip", name:"renamed.zip"}]

};

// send mail with defined transport object

transporter.sendMail(mailOptions, (error, info) => {

if (error) {

return console.log(error);

}

});

});

};

1. Reads the extracted files

exports.upload = function (req, res, next) {

if(req.session.username.substring(0,1) == 'A'){

var array\_name = ['f11\_ar','localarea\_listings\_ar','f12\_ar','f13\_ar','f14\_ar','f15\_ar','f16\_ar','f17\_ar','f21\_ar','f31\_ar','f32\_ar','f41\_ar','f42\_ar','f43\_ar','f44\_ar','f45\_ar','f46\_ar','f47\_ar','f48\_ar','f49\_ar','f410\_ar','f411\_ar','f52\_ar','f53\_ar','f54\_ar','f55\_ar','f56\_ar','f57\_ar','f58\_ar','f59\_ar','f73\_ar','f82\_ar'];

}

if(req.session.username.substring(0,1) == 'T'){

var array\_name = ['f11\_tl','localarea\_listings\_tl','f12\_tl','f13\_tl','f14\_tl','f15\_tl','f16\_tl','f17\_tl','f21\_tl','f31\_tl','f32\_tl','f41\_tl','f42\_tl','f43\_tl','f44\_tl','f45\_tl','f46\_tl','f47\_tl','f48\_tl','f49\_tl','f410\_tl','f411\_tl','f52\_tl','f53\_tl','f54\_tl','f55\_tl','f56\_tl','f57\_tl','f58\_tl','f59\_tl','f73\_tl','f82\_tl'];

}

var array\_length = array\_name.length;

fs.appendFile('logs\_'+req.session.username+'.txt', '['+dateLog+'] Transmitting '+req.file.originalname+' \n');

for(var i = 0 ; i< array\_length ; i++){

var name = array\_name[i]+"\_"+req.file.originalname.substring(0,req.file.originalname.length-8)+".csv";

csvFilePath='C:/dcs/consolidation/'+array\_name[i].substring(0,array\_name[i].length-3)+"\_"+req.file.originalname.substring(0,req.file.originalname.length-8)+".csv";

var form\_name = array\_name[i];

csv\_transmit(name,form\_name);

}

1. Perform passed function

function csv\_transmit(name,form\_name,username){

csv()

.fromFile(csvFilePath)

.on('json',(jsonObj)=>{

JSON2mySQL(jsonObj,name,form\_name,username);

})

.on('done',(error)=>{

console.log('Upload Success');

})

};

1. Insert data per form into FNRI server database

function JSON2mySQL(json,name,form\_name,username){

var col = [];

var values =[];

var sql ="";

var rn = 0;

var arr = name.split('\_');

for (var k in json){

if(k=='id'){

// console.log(sql);

if(col.length>0){

db2.all(sql ,function(err,rows){

if(err) //console.log(err);

rn++;

});

col = [];

values = [];

}

}

else{

col.push(k);

values.push(json[k]);

}

}

        // console.log(form\_name);

if(col.length>0){

sql = "INSERT IGNORE INTO "+form\_name+"(" + col.join(",") + ") VALUES('" + values.join("','") + "');";

connection.query(sql ,function(err,rows){

                    if(err){

                        console.log("Error number : " + err);

                        fs.appendFile('logs\_'+username+'.txt', '['+dateLog+'] Error on transmission on '+form\_name+' : '+err+' \n');

                }

rn++;

                     console.log("INSERTED RECORD : "+form\_name);

});

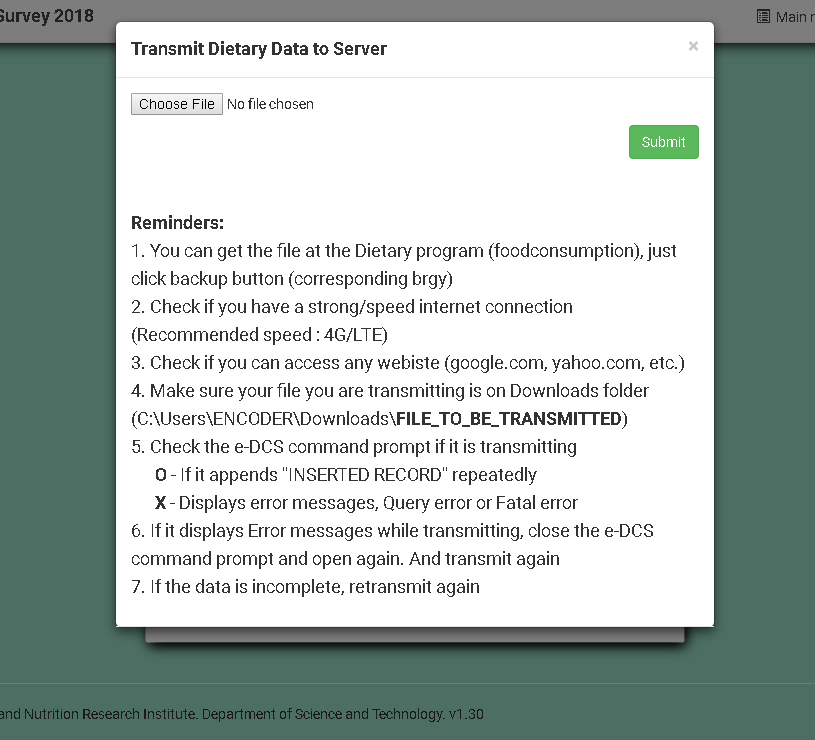
col = [];

values = [];

}

}

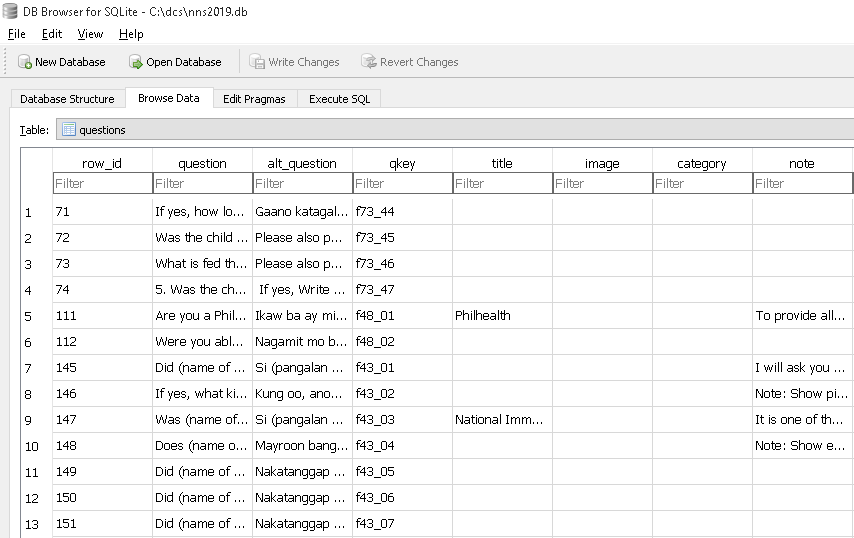
***Sample output (localsurvey.ejs) :***



***Sample output (transmission.ejs) – if the transmission is completed :***

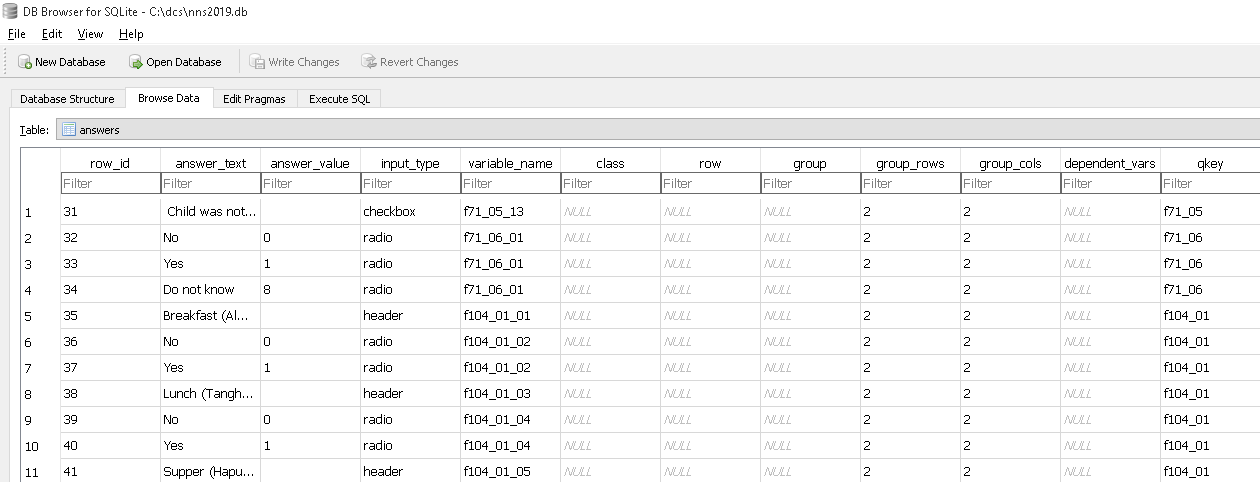
**3.2.10 Form Questions**

This method selects and sends the returned data from questions table (depending on which the user want to enter) on nns2018.db generated by form generator to client-side.



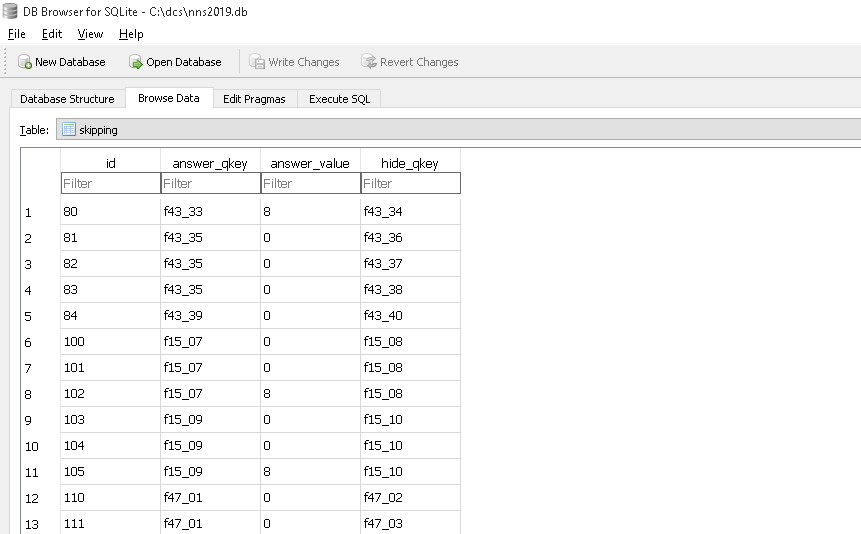
**3.2.11 Form Answers**

This method selects and sends the returned data from answers table on nns2018.db generated by form generator to client-side.



**3.2.12 Form Skipping**

This method selects and sends the returned data from skipping table on nns2018.db generated by form generator to client-side.



***Questions, answers, skipping function (form411.js):***

exports.get = function(req, res){

    var id = req.params.id;

    var eacode2 = id.substring(0, 12);

    var hcn2 = id.substring(12, 16);

    var shsn2 = id.substring(16, 20);

    var MEMBER\_CODE2 = id.substring(20,22);

    var SURNAME2 = id.substring(22,66);

db2.all("SELECT \* FROM f11 WHERE eacode = ? and hcn= ? and shsn= ? and MEMBER\_CODE=?",[eacode2, hcn2, shsn2, MEMBER\_CODE2], function(err,rows){

    db2.all("SELECT \* FROM f411 WHERE eacode = ? and hcn= ? and shsn= ? and MEMBER\_CODE=?",[eacode2, hcn2, shsn2, MEMBER\_CODE2], function(err,rows2){

        db2.all("SELECT \* FROM questions WHERE qkey LIKE '%f411\_%' order by qkey ASC", function(err,rows3){

            db2.all("SELECT \* FROM answers WHERE qkey LIKE '%f411\_%' order by variable\_name ASC", function(err,rows4){

                db2.all("SELECT \* FROM survey\_forms WHERE table\_name='f411'", function(err,rows5){

                    res.render('form411.ejs',{array:rows,array2:rows2,array3:rows3,array4:rows4,array5:rows5,layout:false,session:req.session});

                });

            });

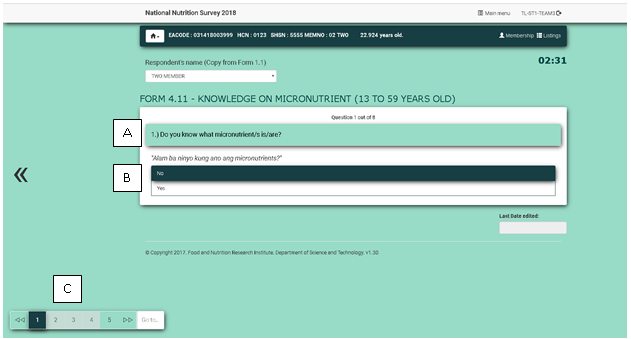
        });

    });

});

};

***Sample output (form411.ejs) – random example :***



Legend:

1. Questions data – ordered by qkey
2. Answers data – ordered by variable\_name
3. Skipping data – will skip answers depending on the answer

**4.0 DATABASE**

**4.0 SYSTEM DATABASE**

Detailed information of tables and structure of nns2018.db database used in the program.

**4.1 Membership form**

Table name: f11

|  |  |
| --- | --- |
| **Id** | Primary key identification |
| **eacode** | Area code |
| **hcn** | Housing control number |
| **shsn** | Sampling household serial number |
| **MEMBER\_CODE** | Member number |
| **Surname** | Last name of individual |
| **Givenname** | First name of individual |
| **Nbi** | Sleep last night |
| **Mom** | Mother of individual |
| **Adopted** | If individual is adopted |
| **Biomom** | If biological mother |
| **Biodad** | If biological father |
| **Dad** | Father of individual |
| **Dbirth** | Date of birth of individual |
| **Refdate** | Reference date of interview |
| **Es\_dbi** | If it is estimated date of birth |
| **Age** | Age of individual |
| **Sex** | Sex code of individual |
| **Csc** | Civil status code of individual |
| **Psc** | Physiological code of individual |
| **Maternal** | If the individual is pregnant |
| **Rhc** | Relation of individual to the household head |
| **Educ** | Education status of individual |
| **Educ\_course** | Education course of individual |
| **Educ\_oth** | Other education course |
| **School** | School attended by the individual |
| **Work** | Work of individual |
| **Occupation** | Occupation of individual |
| **Occupation\_code** | Occupation code of individual |
| **Workplace** | Workplace of individual |
| **W\_class** | Work class of individual |
| **Religion** | Religion of individual |
| **Oth\_rel** | Other religion if not stated |
| **Remarks** | Remarks of individual |
| **Memkey** | Unique key |
| **Date\_added** | Date encoded by the researcher |
| **Date\_edit** | Date edited of individual data by the researcher |
| **Interview\_status** | Interview status of individual |
| **Interview\_status\_oth** | Other interview status |
| **Interview\_status1** | First call back of individual |
| **Interview\_status2** | Second call back of individual |
| **Interview\_status3** | Last call back of individual |
| **Visit1** | Date of first call back of individual |
| **Visit2** | Date of second call back of individual |
| **Visit3** | Date of last call back of individual |
| **Comment1** | remarks of first call back of individual |
| **Comment2** | remarks of second call back of individual |
| **Comment3** | remarks of last call back of individual |
| **Interview\_statusf** | Final interview status of individual |
| **Username** | Interviewer’s username |
| **Is\_f31 ~> is\_82** | Interview status by forms |

**4.2 Listing of households**

Table name: localarea\_listings

|  |  |
| --- | --- |
| **Id** | Primary key identification |
| **eacode** | Area code |
| **hcn** | Housing control number |
| **shsn** | Sampling household serial number |
| **Respondent** | Respondent’s name |
| **Hhead** | Household head’s name |
| **Address** | Address of household |
| **Visit1** | Date of first call back of household |
| **Interview\_status1** | First call back status of household |
| **Visit2** | Date of second call back of household |
| **Interview\_status2** | second call back status of household |
| **Visit3** | Date of Last call back of household |
| **Interview\_status3** | Last call back status of household |
| **Date\_added** | Household encoded date |
| **Interview\_status4** | Final interview status of household |
| **Interview\_status\_oth** | Other remarks of household |
| **Diebio** | Dietary or biochemical household (50%) |
| **Ref\_date** | Replacement household |
| **Date\_edit** | Hcn/shsn of replaced household |
| **F12\_03\_01** | First answer of refused households (observation) |
| **F12\_04\_01** | Second answer of refused households (observation) |
| **F12\_05\_01** | Third answer of refused households (observation) |
| **F12\_06\_01** | Not used |
| **Latitude** | Latitude coordinates of household |
| **Longitude** | Longitude coordinates of household |

**4.3 Household forms**

The columns will depend on the data generated by the form generator, the table below will display only the default columns

Table name: f12, f13, f14, f15, f16, f17

|  |  |
| --- | --- |
| **Id** | Primary key identification |
| **eacode** | Area code |
| **hcn** | Housing control number |
| **shsn** | Sampling household serial number |
| **RES\_NAME** | Respondent name |
| **DATE\_ADDED** | Date form created |
| **DATE\_EDIT** | Date form edited |
| **INTERVIEW\_STATUS** | Interview status of form |
| **INTERVIEW\_TIME** | Interview duration in seconds |
| **TIME\_START** | Interview time and date start |
| **TIME\_END** | Interview time and date end |
| **username** | Interviewer’s username |

**4.4 Individual forms**

The columns will depend on the data generated by the form generator; the table below will display only the default columns

Table name: f21,f31,f32,f41,f42,f43,f44,f45,f46,f47,f48,f49,f410,f411,f52,f53,f54,f55,f56,f57,f58,f59,f73,f82

|  |  |
| --- | --- |
| **Id** | Primary key identification |
| **eacode** | Area code |
| **hcn** | Housing control number |
| **shsn** | Sampling household serial number |
| **MEMBER\_CODE** | Member number |
| **RES\_NAME** | Respondent name |
| **DATE\_ADDED** | Date form created |
| **DATE\_EDIT** | Date form edited |
| **INTERVIEW\_STATUS** | Interview status of form |
| **INTERVIEW\_TIME** | Interview duration in seconds |
| **TIME\_START** | Interview time and date start |
| **TIME\_END** | Interview time and date end |
| **username** | Interviewer’s username |

**4.5 Answers**

Table name: answers

|  |  |
| --- | --- |
| **row\_id** | Primary key identification |
| **Answer\_text** | e.g. Yes, No |
| **Answer\_value** | e.g. 1, 0 |
| **Input\_type** | e.g. Radio, checkbox, text |
| **Variable\_name** | e.g. f12\_01\_01 |
| **Class** | Not Used |
| **Row** | Not Used |
| **Group** | Not Used |
| **Group\_rows** | Not Used |
| **Group\_cols** | Not Used |
| **Dependent\_vars** | Not Used |
| **Qkey** | Form + Answer number e.g. f12\_01 |
| **Nextq** | Not Used |
| **Skip** | Not Used |
| **attrib** | Not Used |

**4.6 Questions**

Table name: questions

|  |  |
| --- | --- |
| **Row\_id** | Primary key identification |
| **Question** | Question in English |
| **Alt\_question** | Question in Tagalog |
| **Qkey** | Form + Question number e.g. f12\_01 |
| **Title** | Title or introduction in forms |
| **Image** | Not Used |
| **Category** | Not Used |
| **Note** | Note in forms usually in bold |
| **Others** | Not Used |
| **Required** | Not Used |
| **Tabular** | Not Used |
| **Rows** | Not Used |
| **Cols** | Not Used |
| **colheader** | Not Used |

**4.8 Skipping**

Table name: skipping

|  |  |
| --- | --- |
| **Id** | Primary key identification |
| **Answer\_qkey** | Answer key |
| **Answer\_value** | e.g. 1,0 |
| **Hide\_qkey** | Skipped questions |

**4.9 localsurveyareas**

Table name: localsurveyareas

|  |  |
| --- | --- |
| **Row\_id** | Primary key identification |
| **Region** | Region code |
| **Province** | Province code |
| **Municipality** | Municipality code |
| **Barangay** | Barangay code |
| **Ea** | 3 digit code |
| **Areaname** | Area name |
| **Eacode** | Area code |
| **Remarks** | Not Used |
| **Date\_start** | Not Used |
| **Date\_end** | Not Used |
| **Team** | Not Used |
| **Subteam** | Not Used |

**4.10 login**

Table name: login

|  |  |
| --- | --- |
| **id** | Primary key identification |
| **name** | Not Used |
| **username** | Researcher position + subteam + team |
| **Type** | 4 character password |
| **Password** | Not Used |
| **category** | Not Used |