Image Processing and Data Visualization for
the Public Health Monitoring System of
Filipino Children

Technical Manual
Presented to
the Faculty of the College of Computer Studies
De La Salle University

In Partial Fulfillment
of the Requirements for the Degree of
Bachelor of Science in Computer Science

by

Cardano, Marc Daniel T.
Dy, Stephanie Joy D.
Lim, Reanna Chelsey N.
Pagtalunan, Dominic M.

Judith J. Azcarraga, PhD
Adviser

November 20, 2017

Table of Contents

I. Introduction

This is the technical manual for the Public Health Monitoring System of Filipino Children under GetBetter Telemedicine System in the Philippines. The aim of this system is to aid the routine wellness checks for children in communities which may not have accessible, affordable, or quality health care.

II. System Design

The improved Public Health Monitoring System consists of four major components: the data collection application (GeeBee), the image processing application (GeeBeeCapture), the data visualization application (GeeBeeView) and the cloud hosted database. GeeBee, GeeBeeCapture, and GeeBeeView are mobile applications designed for tablets while the cloud database is a DigitalOcean droplet. GeeBee already existed prior to this research developed by Lacsamana et al. (2016) that can perform basic monitoring and consultation activities. It had some minor improvements such as connecting its local database to the cloud for uploading and an interrelated process with GeeBeeCapture but the process flow is almost unchanged. GeeBeeCapture was developed in this research to calculate height and weight using image processing and returns the calculated values back to GeeBee as an alternative for manual measuring. GeeBeeView was also created in this research to display the collected data from GeeBee. The cloud hosted database is nothing more than a storage space in the cloud so that sharing data is more efficient.

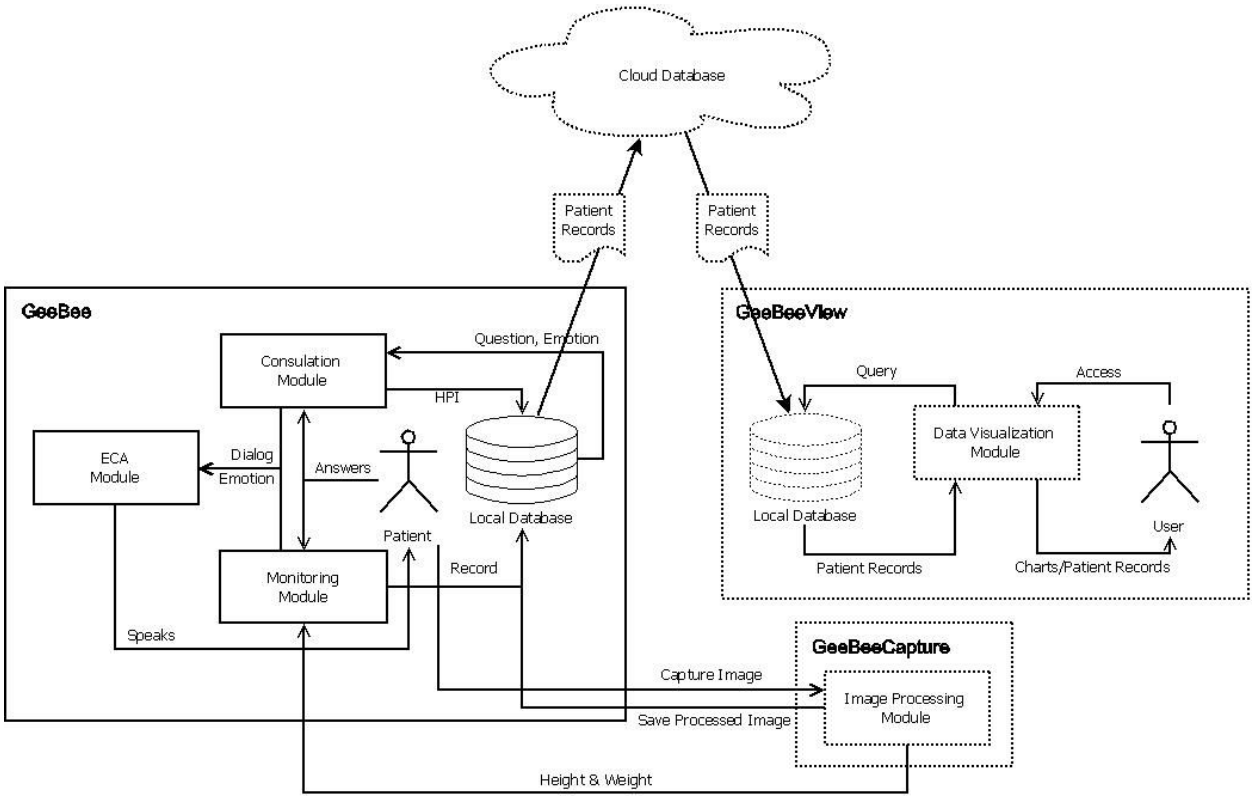

Figure 2.1 System Architecture of Public Health Monitoring System

III. GeeBee

GeeBee is the mobile application for collecting and recording data and consultation records of patients. It is also tasked to upload the data to the database. There were two main additions in this preexisting app, the uploading and the selection of schools. Since this was already a made app by a previous theisis group, the proponents only add classes and such that was modified or added by the proponents.

A. Class Dictionary

PATH	/src/main/java/com.geebeelicious.geebeelicious/activities/SettingsActivity.java
CLASS	SettingsActivity
SUPERCLASS	ActionVarActivity
PROPERTIES	1. schoolsAdapter -- The schools adapter that handles the data shown on the RecyclerView 2. schoolsRecyclerView -- The RecyclerView that shows the available Schools. 3. schoolLayoutManager -- The layout manager that keeps the constraints of the recyclerView 4. addSchoolButton -- The button that lets the user add a new School. 5. removeSchoolButton -- The button that lets the user remove a new school. 6. URL_SAVE_NAME - url of server to access. 7. geebeeDb -- the database adapter for easy database use. 8. chosenSchool -- the school to use. 9. uploadAllPatientData -- button that initializes upload.
METHODS	1. onCreate() -- builds the UI of the Activity. 2. saveSchool() -- saves the selected school to use in the session. 3. addChooseSchoolSetting() -- adds a school in the list 4. uploadAllData() -- searches for flagged data that aren't synced yet and request them to the server as POST by using postRequestUpload(). 5. postRequestUpload(String typeRequest,String jsonToSend) -- sends volley request to server.

PATH	/src/main/java/com.geebeelicious.geebeelicious/adapters/RecyclerViewClickListener.java
CLASS	RecyclerViewClickListener
SUPERCLASS	N/A (Interface)
PROPERTIES	N/A (Interface)
METHODS	N/A (Interface)

PATH	/src/main/java/com.geebeelicious.geebeelicious/activities/SchoolsRecyclerAdapter.java
CLASS	SchoolsRecyclerAdapter
SUPERCLASS	RecyclerView.Adapter<SchoolsRecyclerAdapter.SchoolViewHolder>
PROPERTIES	1. ArrayList<School> schools -- list of schools in the recyclerview. 2. Context schoolContext -- context of the activity. 3. RecyclerViewClickListener itemListener -- the click listener for the view holders. 4. int selectedIndex -- the selected position in the recyclerView.
METHODS	1. SchoolsRecyclerAdapter(Context context, ArrayList<School> schools, RecyclerViewClickListener itemListener) -- Instantiates the class on call. 2. onCreateViewHolder(ViewGroup parent, int viewType) -- initializes the viewholder. 3. onBindViewHolder(SchoolViewHolder holder, int position) -- binds the inner elements of each row in list. 4. getSelectedIndex() -- return selected index in list. 5. getitemCount() -- returns the size of list of schools.

PATH	/src/main/java/com.geebeelicious.geebeelicious/activities/SchoolsRecyclerAdapter.java (inner class of SchoolsRecyclerAdapter)
CLASS	SchoolViewHolder
SUPERCLASS	extends RecyclerView.ViewHolder implements View.OnClickListener
PROPERTIES	1. TextView tv_schoolName -- the UI text for showing school name in view holder. 2. TextView tv_municipalityName -- the UI text for showing municipality in view holder.
METHODS	1. SchoolViewHolder(View itemView) -- Instantiates the view holder. 2. onClick(View v) -- shows a toast on what the user has selected.

PATH	/src/main/java/com.geebeelicious.geebeelicious/databas/DatabaseAdapter.java
CLASS	DatabaseAdapter
SUPERCLASS	N/A

PROPERTIES	1. Context context -- the context of Database Adapter.
METHODS	1. getAllSchools() -- return list of school retrieved from the database. 2. getPatientsFromSchool(int schoolID) -- gets arraylist of patients from a school. 3. getAllUnsyncedRows() -- gets all rows from database that has not been uploaded yet. Returns Syncable type. 4. insertSchool(School school) -- insert a school to the database. 5. getAllRegions() -- gets a list of all the regions. Returns arraylist of string. 6. getAllProvincesFrom(int regionId) -- gets all provinces that resides in the region in regionId. Returms arraylist of string. 7. getAllMunicipalitiesFrom(int provinceId) -- gets all municipalities residing in province of provinceId. Return arraylist of String. 8. getProvinceId(String provinceName) -- gets the id of the corresponding provinceName. Returns int. 9. getRegionId(String regionName) -- retruns the corresponding region id from a region name. Retuns int. 10. removeSchool(int id) -- deletes a school from the database by using the id. Returns nothing (void). 11. getMunicipality(int municipalityId) -- gets municipality name from an its corresponding id. 12. setToSynced() -- sets all the synced columns to synced. To be used once upload is confirmed. Returns boolean.

PATH	/src/main/java/com.geebeelicious.geebeelicious/databas/DatabaseAdapter.java
CLASS	DatabaseAdapter
SUPERCLASS	N/A
PROPERTIES	1. List<Patient> unsyncedPatients -- all the unsynced patients in a list. 2. List<Record> unsyncedRecords -- all the unsynced records in a list. 3. List<School> unsyncedSchool -- all the unsynced schools in list;
METHODS	1. getUnsyncedPatients() -- get all unsynced patients. Returns arraylist of patients. 2. getUnsyncedRecords() -- get all unsynced records. Returns arraylist of records. 3. getUnsyncedSchools() -- get all unsynced schools. Returns arraylist of schools. 4. getPatientJSON() -- get the JSON representation of patient for sending to remote database. Returns String. 5. getRecordJSON() -- get the JSON representation of record for sending to remote database. Returns String. 6. getSchoolJSON() -- get the JSON representation of school for sending to remote database. Returns String.
NOTES	This class uses third party library from Google called “Gson”. A library that can encode objects to JSON and decode JSON. https://github.com/google/gson

B. Database Dictionary

a. Remote Database

Table	Description
Province	Reference for the 81 provinces of the Philippines `id: internal id of province` `regCode:` region where province is located `provDesc:` name of province provCode: provincial geographic code (PSGC, 2017) NOTE: modified from https://github.com/clavearnel/philippines-region-province-citymun-brgy
Municipality	Reference for the 1647 municipalities of the Philippines `citymunCode`:identification for municipality `provCode:` foreign key for province where municipality is located `citymunDesc:` municipality name regDesc: region id foreign key NOTE: modified from https://github.com/clavearnel/philippines-region-province-citymun-brgy
Region	Reference for the 17 regions of the Philippines Id: internal id of region regDesc: region name regCode: code of region NOTE: modified from https://github.com/clavearnel/philippines-region-province-citymun-brgy
School	Reference for school `school_id:` identification for school `citymunCode:` foreign key for municipality where school is located `name:` name of school
Patient	Patient information and details `patient_id:` identification for patient `school_id:` foreign key for school where patient is enrolled `first_name:` first name of patient `last_name:` last name of patient `birthday:` date of birth `gender:` sex (0 - male; 1 - female) `handedness:` dominant hand (0 - right; 1 - left) `remarks_string:` remarks about patient `remarks_audio:` audio recorded remarks
Record	Monitoring records generated by monitoring activities `record_id:` identification for record `patient_id:` foreign key for patient who conducted monitoring activity `date_created:` date when monitoring activity was conducted `height:` height `weight:` weight (20/200, 20/100, 20/70, `visual_acuity_left:` visual acuity of left eye 20/50, 20/40, 20/30, 20/25, `Visual_acuity_right:` visual acuity of right eye 20/20, 20/15, 20/10, 20/5)

	color_vision: color vision (Normal, Abnormal) hearing_left: hearing of left ear (Normal Hearing, Mild Hearing Loss, Moderate hearing_right: hearing of right ear Hearing Loss, Moderately-Severe Hearing) gross_motor: gross motor (0 - Pass; 1 - N/A; 2 - Fail) gross_motor_remarks: remarks on gross motor test fine_motor_dominant: fine motor dominant hand (0 - Pass; 1 - Fail) fine_motor_n_dominant: fine motor non-dominant hand fine_motor_hold: fine motor hold (0 - Hold: 1 - Not Hold) vaccination: picture of immunization record patient_picture: picture of patient when monitoring activity was conducted remarks_string: remarks about patient remarks_audio: audio recorded remarks
Dataset	Groupings made to section data using school and date of creation as basis dataset_id: identification for dataset school_id: foreign key for school where data was collected date_created: foreign key when data was collected

b. Local Database

Table	Description
Province	Reference for the 81 provinces of the Philippines id: internal id of province regCode: region where province is located provDesc: name of province provCode: provincial geographic code (PSGC, 2017) NOTE: modified from https://github.com/clavearnel/philippines-region-province-citymun-brgy
Municipality	Reference for the 1647 municipalities of the Philippines citymunCode:identification for municipality provCode: foreign key for province where municipality is located citymunDesc: municipality name regDesc: region id foreign key NOTE: modified from https://github.com/clavearnel/philippines-region-province-citymun-brgy
Region	Reference for the 17 regions of the Philippines Id: internal id of region regDesc: region name regCode: code of region NOTE: modified from https://github.com/clavearnel/philippines-region-province-citymun-brgy
School	Reference for school school_id: identification for school citymunCode: foreign key for municipality where school is located name: name of school synced: synced flag for uploading

Patient	Patient information and details `patient_id`: identification for patient `school_id`: foreign key for school where patient is enrolled `first_name`: first name of patient `last_name`: last name of patient `birthday`: date of birth `gender`: sex (0 - male; 1 - female) `handedness`: dominant hand (0 - right; 1 - left) `remarks_string`: remarks about patient `remarks_audio`: audio recorded remarks synced: synced flag for uploading
Record	Monitoring records generated by monitoring activities `record_id`: identification for record `patient_id`: foreign key for patient who conducted monitoring activity `date_created`: date when monitoring activity was conducted `height`: height `weight`: weight (20/200, 20/100, 20/70, `visual_acuity_left`: visual acuity of left eye 20/50, 20/40, 20/30, 20/25, `Visual_acuity_right`: visual acuity of right eye 20/20, 20/15, 20/10, 20/5) `color_vision`: color vision (Normal, Abnormal) `hearing_left`: hearing of left ear (Normal Hearing, Mild Hearing Loss, Moderate `hearing_right`: hearing of right ear Hearing Loss, Moderately-Severe Hearing) `gross_motor`: gross motor (0 - Pass; 1 - N/A; 2 - Fail) `gross_motor_remarks`: remarks on gross motor test `fine_motor_dominant`: fine motor dominant hand (0 - Pass; 1 - Fail) `fine_motor_n_dominant`: fine motor non-dominant hand `fine_motor_hold`: fine motor hold (0 - Hold: 1 - Not Hold) `vaccination`: picture of immunization record `patient_picture`: picture of patient when monitoring activity was conducted `remarks_string`: remarks about patient `remarks_audio`: audio recorded remarks synced: synced flag for uploading
Dataset	Groupings made to section data using school and date of creation as basis `dataset_id`: identification for dataset `school_id`: foreign key for school where data was collected `date_created`: foreign key when data was collected

IV. GeeBeeCapture

GeeBeeCapture is a mobile application that is tasked to calculate an estimated height and weight of a child based on an image. It takes two images, namely, the front view and the side view of the child and converts it into a silhouette. It then calculates for an estimated height and weight and saves the silhouette image in the database. The height and weight values are then returned to the GeeBee application.

A. Libraries

OpenCv Library Version 3.2

OpenCV (Open Source Computer Vision Library) is the open source core library used for computer vision, image processing and machine learning. Originally developed by Intel, it was later supported by Willow Garage and is now maintained by Itseez

The following package was download and used from the opencv website:

`opencv-3.2.0-vc14.exe`

B. Class Dictionary

This section contains activities with their respective xml files. Thus, they override the methods `onCreate(Bundle)`.

PATH	layout/tablet_layout.java
CLASS	`MainActivityTablet`
SUPERCLASS	`AppCompatActivity`
PROPERTIES	`String TAG: identifier for the source of a log message` `Int CameraRequest1` – identifier for onactivity result for camera request of first picture `Int CameraRequest2` – identifier for onactivity result for camera request of second picture `Int SELECT_PHOTO1` – identifier for intent of first picture `Int SELECT_PHOTO2` – identifier for intent of second picture `Int currentImage` – identifies which image is being processed(0 for front picture, 1 for the side picture) `ImageView imageView1` – variable for imageView for front picture `ImageView imageView2` – variable for imageView for side picture `Double HEIGHT` – height real world measurement of reference object `Button btn_take_picture1` – variable for taking the front view picture `Button btn_load_image1` – variable for loading the front view picture from storage. `Button btn_take_picture2` – variable for taking the side view picture. `Button btn_load_image2` – variable for loading the side view picture from storage. `Button btn_compute` – variable for computing the height and weight of the image `Button btn_go_back` – variable for going back the user flow. `Button btn_cancel` – cancels the intent `TextView tv_height` – contains the resulted height of the subject `TextView tv_weight` – contains the resulted weight of the subject.

METHODS	`SaveImage(Bitmap) :void` – saves the processed photo
	`instantiateVariables():void` - instantiates the required variables
	`onActivityResult(int requestCode, int resultCode, Intent data):void`
	`Can be one of the following:`

	- displays the image taken from camera or loaded from gallery to the imageview - calls on IPUtils to calculate Image getRealPathFromURI(Uri contentUri): String addImageToGallery

PATH	layout/IPUtils.java
CLASS	`IPUtils.java`
SUPERCLASS	
PROPERTIES	
METHODS	`cropImage(Mat mat): Mat` - crops the image to retain the subject and reference object at the center. `basicTresholdedForView(Mat mat): Bitmap` - thresholds the image that will be saved to the database. `basicThresholdAlgo(Mat imageMat):Mat` - applies the following opencv functions on the image: ● GaussianBlur ● Threshold ● Canny ● Dilate ● erode `HeightAndWeightAlgorithm(double realWorldHeight,double lowestCoordinate1,lowestCoordinate2,Mat frontImage ,Mat sideImage):HWBiometrics` - Applies the height and weight algorithm to extract the height and weight of subject, returns HWBiometrics. `getReferenceAndSubjectHeightAndBiggestWidth(List<MatOfPoint> Contours):double[] numbers` -loops over the contours one by one, takes 3 numbers: ● HighestCoordinate X ● LeftMostContour's pixel height ● BiggestContourWidth

PATH	
CLASS	`HWBiometrics.java`
SUPERCLASS	
PROPERTIES	`Double height` - variable containing height; `Double weight`- variable containing weight
METHODS	

PATH	
CLASS	Constants.javan
SUPERCLASS	
PROPERTIES	String HEIGHT - final string contains 'height' String WIDTH - final string contains 'width' String HeightOrWidth - determines if width or height of reference object.
METHODS	

V. GeeBeeView

GeeBeeView is a mobile application that is tasked to display the data collected by GeeBee. It downloads the uploaded data from the shared cloud database and stores it in a local database. After downloading the data downloaded can be viewed from this application in the form of charts or as raw data.

A. Database

GeeBeeView has its own local database from where the displayed data must first be inserted once downloaded (see Figure 5.1 and Table 5.1).

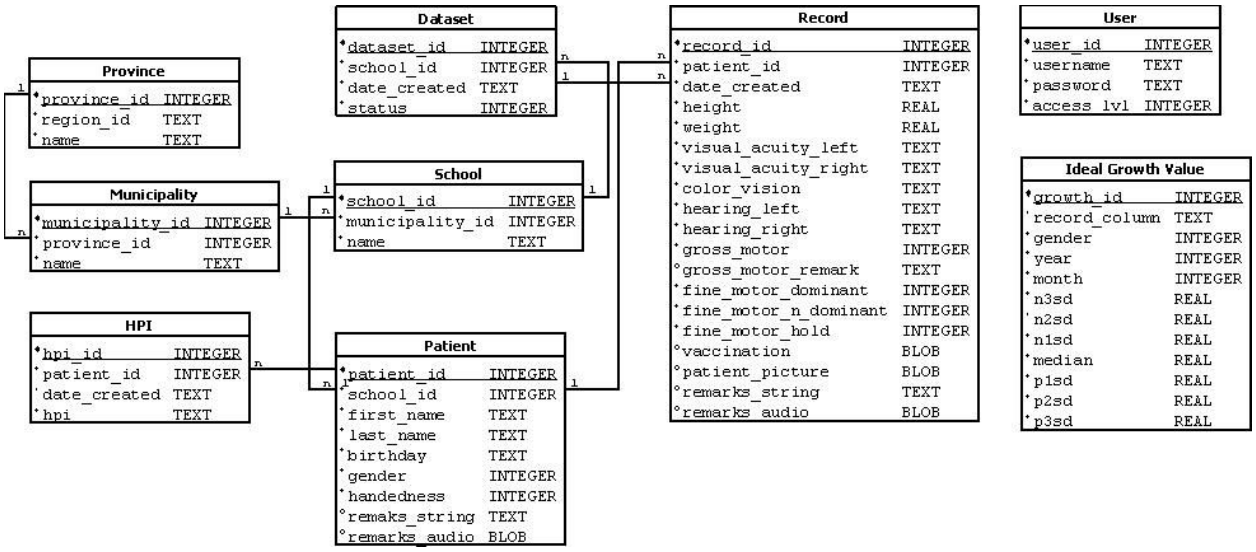

Figure 5.1 Database Table Relational Diagram of Local Database of GeeBeeView

For the purpose of downloading the data from the cloud database, which is a DigitalOcean droplet, the

Dependency, `compile 'com.android.volley:volley:1.0.0'` adds the volley dependency on the application thus enabling the application to be able to communicate with servers hosted on the internet. Once the Volley Singleton Class has been set up, one can send requests and receive responses from servers. Using this model, the way GeeBeeView downloads data from the Database is to first request the dataset to be downloaded. The request mainly contains the request type, the school and the date of the dataset. Once the cloud Server receives this request, it will execute a MySQL query from the parameters given from the request. Once the query is done the result is then sent back to the application as a response and the application will parse the response and integrate the downloaded data to its local database.

Table 5.1 Description of Tables in GeeBeeView

Table	Description
Province	Reference for the 81 provinces of the Philippines `province_id`: provincial geographic code (PSGC, 2017) `region_id`: region where province is located `name`: name of province
Municipality	Reference for municipality `municipality_id`: identification for municipality `province_id`: foreign key for province where municipality is located `name`: municipality name
School	Reference for school `school_id`: identification for school `municipality_id`: foreign key for municipality where school is located `name`: name of school
Patient	Patient information and details `patient_id`: identification for patient `school_id`: foreign key for school where patient is enrolled `first_name`: first name of patient `last_name`: last name of patient `birthday`: date of birth `gender`: sex (0 - male; 1 - female) `handedness`: dominant hand (0 - right; 1 - left) `remarks_string`: remarks about patient `remarks_audio`: audio recorded remarks
HPI (History of Present Illness)	Consultation records generated by consultation activities `hpi_id`: identification for HPI `patient_id`: foreign key for patient who conducted consultation activity `date_created`: date when consultation activity was conducted `hpi`: HPI generated listing symptoms
Record	Monitoring records generated by monitoring activities `record_id`: identification for record `patient_id`: foreign key for patient who conducted monitoring activity `date_created`: date when monitoring activity was conducted

	height: height weight: weight (20/200, 20/100, 20/70, visual_acuity_left: visual acuity of left eye 20/50, 20/40, 20/30, 20/25, Visual_acuity_right: visual acuity of right eye 20/20, 20/15, 20/10, 20/5) color_vision: color vision (Normal, Abnormal) hearing_left: hearing of left ear (Normal Hearing, Mild Hearing Loss, Moderate hearing_right: hearing of right ear Hearing Loss, Moderately-Severe Hearing) gross_motor: gross motor (0 - Pass; 1 - N/A; 2 - Fail) gross_motor_remarks: remarks on gross motor test fine_motor_dominant: fine motor dominant hand (0 - Pass; 1 - Fail) fine_motor_n_dominant: fine motor non-dominant hand fine_motor_hold: fine motor hold (0 - Hold: 1 - Not Hold) vaccination: picture of immunization record patient_picture: picture of patient when monitoring activity was conducted remarks_string: remarks about patient remarks_audio: audio recorded remarks
Dataset	Groupings made to section data using school and date of creation as basis dataset_id: identification for dataset school_id: foreign key for school where data was collected date_created: foreign key when data was collected status: status of dataset in local database (0 - Not Downloaded; 1 - Downloaded)
Ideal Growth Value	Standard growth values based on growth reference chart tables for children in ages 5 to 19 (WHO, 2007) growth_id: identification for ideal growth record record_column: description of which body measurement gender: gender of patient year: age of patient in years month: age of patient in months added after year n3sd: growth value of -3 standard deviations (SD) from standard n2sd: growth value of -2 SD from standard n1sd: growth value of -1 SD from standard median: standard growth value p1sd: growth value of 1 SD from standard p2sd: growth value of 2 SD from standard p3sd: growth value of 3 SD from standard
User	User information user_id: identification for user username: identification for user password: password access_lvl: access level of user (0 - No access; 1 - Allowed access)

B. Chart Generation

The android library MPAndroidChart, developed by Philipp Jahoda (PhilJay), was used for offline chart generation. The latest jar file available at the time of development was added as a library.

mpandroidchartlibrary-2-2-4.jar

C. Class Dictionary

C.1. Adapters

The adapter classes in this section are `RecyclerView.Adapter`. Thus, all these classes will automatically contain and override the inherited methods to fit their needs specifically the methods: `onCreateViewHolder(ViewGroup, int)`, `onBindViewHolder(RecyclerView.ViewHolder, int)`, and `getItemCount()`.

PATH	adapter/DatasetAdapter.java
USAGES	`DatasetListActivity`
CLASS	`DatasetAdapter`
SUPERCLASS	`RecyclerView.Adapter<DatasetAdapter.DatasetViewHolder>`
PROPERTIES	`String URL_SAVE_NAME`: reference to the PHP page to where cloud database is connected `String TAG`: reference to class for tracing error using log `ArrayList<Dataset> datasetList`: content of the recyclerview `Context context`: context of the activity `DatabaseAdapter getBetterDb`: adapter for database connection

METHODS	`DatasetAdapter(ArrayList<Dataset> datasetList)` Constructor
	`openActivity(Dataset dataset)` stores `dataset` as `Intent.Extra` then opens `DataVisualizationActivity`
	`downloadDataset(Dataset dataset)` downloads `dataset` when "DOWNLOAD" button is clicked
	`clearDatasetList()` clear `datasetList`

SUBCLASS	`DatasetViewHolder`
SUPERCLASS	`RecyclerView.ViewHolder`
PROPERTIES	`TextView tvSchoolName`: view where school name of a dataset is displayed `TextView tvDate`: view where date of a dataset is displayed `Button btnStatus`: view where status of a dataset is displayed
METHODS	`DatasetViewHolder(View view)` Constructor

PATH	adapter/HPIListAdapter.java
USAGES	`HPIListActivity`
CLASS	`HPIListAdapter`
SUPERCLASS	`RecyclerView.Adapter<HPIListAdapter.HPIListViewHolder>`
PROPERTIES	`ArrayList<HPI> HPIList`: content of the recyclerview

	`Context context:` context of the activity
METHODS	`HPIListAdapter(ArrayList<HPI> HPIList)` Constructor `getPatientFromDB(int patientID): Patient` get the patient information
SUBCLASS	`HPIListViewHolder`
SUPERCLASS	`RecyclerView.ViewHolder`
PROPERTIES	`TextView tvPatientName:` view where patientl name is displayed `TextView tvDate:` view where date of consultation is displayed `Button btnView:` view to open `ViewHPIActivity`
METHODS	`HPIListViewHolder(View view)` Constructor

PATH	adapter/PatientListAdapter.java
USAGES	`PatientListActivity`
CLASS	`PatientListAdapter`
SUPERCLASS	`RecyclerView.Adapter<PatientListAdapter.PatientListViewHolder>`
PROPERTIES	`ArrayList<Patient> patientList:` content of the recyclerview `Context context:` context of the activity `String recordDate:` date of dataset
METHODS	`PatientListAdapter(ArrayList<Patient> patientList)` Constructor
SUBCLASS	`PatientListViewHolder`
SUPERCLASS	`RecyclerView.ViewHolder`
PROPERTIES	`TextView tvPatientName:` view where patient name is displayed `TextView tvGender:` view where sex of patient is displayed `TextView tvAge:` view where age of patient is displayed `Button btnView:` view to open `ViewPatientActivity`
METHODS	`PatientListViewHolder(View view)` Constructor

PATH	adapter/TextHolderAdapter.java
USAGES	Adapter for dataset list in `DataVisualizationActivity`
CLASS	`TextHolderAdapter`
SUPERCLASS	`RecyclerView.Adapter<TextHolderAdapter.TextAdapterViewHolder>`
PROPERTIES	`String TAG:` used for tracing error in logs `ArrayList<String> textList:` content of the recyclerview `TextListener mListener:` interface to return actions to activity

METHODS	`TextHolderAdapter(ArrayList<String> filterList, TextListener mListener)` Constructor
SUBCLASS	`TextAdapterViewHolder`
SUPERCLASS	`RecyclerView.ViewHolder`
PROPERTIES	`TextView tvDataset:` view where dataset is displayed `TextView tvDatasetClose:` view that acts as the remove filter button
METHODS	`TextAdapterViewHolder(View view)` Constructor
INTERFACE	`TextListener`
METHODS	`removeDataset(String filter)` removes `filter` from the visualization in `DataVisualizationActivity`

PATH	adapter/FilterAdapter.java
USAGES	Adapter for filter list in `DataVisualizationActivity`
CLASS	`FilterAdapter`
SUPERCLASS	`RecyclerView.Adapter<FilterAdapter.FilterAdapterViewHolder>`
PROPERTIES	`String TAG:` used for tracing error in logs `ArrayList<String> filterList:` content of the recyclerview `FilterAdapterListener mListener:` interface to return actions to activity
METHODS	`FilterAdapter(ArrayList<String> filterList, FilterAdapterListener mListener)` Constructor
SUBCLASS	`FilterAdapterViewHolder`
SUPERCLASS	`RecyclerView.ViewHolder`
PROPERTIES	`TextView tvFilterText:` view where filter is displayed `TextView tvFilterClose:` view that acts as the remove filter button
METHODS	`FilterAdapterViewHolder(View view)` Constructor
INTERFACE	`FilterAdapterListener`
METHODS	`removeFilter(String filter)` - removes `filter` from the visualization in `DataVisualizationActivity`

C.2. Database

The classes in this section are for accessing the local database and downloading from the cloud database.

PATH	database/DatabaseAdapter.java

CLASS	DatabaseAdapter
PROPERTIES	String TAG: used for tracing error in logs SQLiteDatabase getBetterDB: database of the app DatabaseHelper getBetterDatabaseHelper: helper class of the database
METHODS	DatabaseAdapter(Context context) - Constructor createDatabase(): DatabaseAdapter - Creates the database using the helper class and returns a reference of itself openDatabseForRead(): DatabaseAdapter - Opens the database for read or write unless the database problem occurs that limits the user from writing to the database and returns a reference of itself openDatabaseForWrite(): DatabaseAdapter - Opens the database for read or write. Method call may fail if a database problem occurs and returns a reference of itself closeDatabase() - closes the database getRecords(int patientID): ArrayList<Record> - get and return all of the records with the patientID getHPIs(int patientID): ArrayList<HPI> - get and return all of the HPIs of the patientID getPatientsFromSchool(int schoolID): ArrayList<Patient> - get and return all patients with the schoolID insertUser(User user): int - insert user to database and return userID getUser(String username): User - get and return the user with the username getRecordsFromSchool(int schoolID, String date): ArrayList<PatientRecord> - get and return all the records with the schoolID and date getPatient(int patientID): Patient - get patient with the patientID getHPIsFromSchool(int schoolID): ArrayList<HPI> - get and return all HPI with schoolID updateDatasetList(Dataset dataset): boolean - insert/update dataset table updateRecord(Record record) - insert/update record table updateDatasetStatus(Dataset) - update dataset status when downloaded updatePatient(Patient patient) - insert/update patient table updateSchools(School school) - insert/update school table getSchoolName(int schoolID): String - get and return school name with schoolID getPatientsWithCondition(int schoolID, String date, String column, String value): ArrayList<Patient> - get and return patients with certain conditions getIdealValue(String recordColumn, int gender, int age): IdealValue - get and return IdealValue of patients with gender, age and recordColumn getAverageRecord(Patient patient, String recordDate): Record - get and return average record with the same age as patient getMostFrequentValue(String birthYear, String recordYear, String recordColumn): Cursor - get and return cursor for every column of table record table

PATH	database/DatabaseHelper.java

CLASS	`DatabaseHelper`
SUPERCLASS	`SQLiteOpenHelper`
PROPERTIES	`String TAG`: identifier for the source of a log message `String DB_PATH`: directory path of the database `String DB_NAME`: name of the database `int DB_VERSION`: version number of database `Context myContext`: context of the database `SQLiteDatabase getBetterDatabase`: database of the application
METHODS	`DatabaseHelper(Context context)` – Constructor `createDatabase()` – creates database `checkDatabase(): boolean` – check whether application database exists `copyDatabase()` – copy database from assets folder `openDatabase()` – opens database

PATH	database/VolleySingleton.java
CLASS	`VolleySingleton`
PROPERTIES	`VolleySingleton mInstance`: instance of itself `RequestQueue mRequestQueue`: request queue to broadcast `Context mCtx`: context of activity
METHODS	`VolleySingleton(Context context)` – Constructor `getInstance(Context context): VolleySingleton` – creates an instance of itself `addToRequestQueue(Request<T> req)` – check whether application database exists
Note: This class was written by Belial on 21/09/16 and not by the proponents	

C.3. Dialogs

This section includes the dialogs and have their respective xml files. Automatically, they override the classes `onCreateDialog(Bundle)` and `onAttach(Activity)`.

PATH	layout/AddDatasetDialogFragment.java
CLASS	`AddDatasetDialogFragment`
SUPERCLASS	`DialogFragment`
PROPERTIES	`Spinner spAddDataset`: dropdown menu to select dataset to be added `ArrayList<Dataset> datasetList`: list of all available datasets `int selectedDatasetIndex`: index of selected dataset from spinner `AddDatasetDialogListener`: interface to return value to activity
METHODS	`prepareSpinner()` – prepare the content and display of the spinner

INTERFACE	`AddDatasetDialogListener`
METHODS	`onDialogPositiveClick(AddDatasetDialogFragment dialog)` – inform activity dialog has chosen to add a dataset

PATH	layout/AddDatasetDialogFragment.java
CLASS	`AddDatasetDialogFragment`
SUPERCLASS	`DialogFragment`
PROPERTIES	`String TAG`: identifier for the source of a log message `Spinner spAgeFilter`: dropdown menu to select equator from the set = {<, <=, =, >=, >} `EditText etFilter`: for numerical age input `String ageValue, ageEquator, genderValue`: values to be returned to activity `AddFilterDialogListener`: interface to return value to activity
INTERFACE	`AddFilterDialogListener`
METHODS	`onDialogPositiveClick(AddFilterDialogFragment dialog)` – inform activity dialog has added filters

C.4. Activities

This section contains activities with their respective xml files. Thus, they override the methods `onCreate(Bundle)`.

PATH	layout/DatasetListActivity.java
CLASS	`DatasetListActivity`
SUPERCLASS	`AppCompatActivity`
PROPERTIES	`String TAG`: identifier for the source of a log message `String URL_SAVE_NAME`: url of PHP page where cloud database is connected `TextView tvTitle, tvSchool, tvDate`: display labels `RecyclerView rvDataset`: list of datasets `ArrayList<Dataset> datasetList`: content of recyclerview `DatasetAdapter datasetAdapter`: adapter for recyclerview `DatabaseAdapter getBetterDb`: access to database
METHODS	`prepareDatasetList()` – get and display dataset list from database `updateDatasets()` – access dataset list from cloud database and update if there is internet connection `updatePreliminary()` – access municipality and school tables from cloud database and update `updateSchoolTable()` – access school table from cloud database and update

PATH	layout/DataVisualizationActivity.java
CLASS	`DataVisualizationActivity`
SUPERCLASS	`AppCompatActivity`

IMPLEMENTS	For dialogs:
	`AddFilterDialogFragment.AddFilterDialogListener`
	`AddDatasetDialogFragment.AddDatasetDialogListener`
	For recylerviews:
	`FilterAdapter.FilterAdapterListener`
	`TextHolderAdapter.TextListener`

PROPERTIES	`String TAG`: identifier for the source of a log message
	`ArrayList<String> datasetList, filterList`: content of recyclerviews
	`TextHolderAdapter datasetAdapter`: adapter for dataset recyclerview
	`FilterAdapter filterAdapter`: adapter for filter recyclerview
	`TextView tvTitle, tvDataset, tvFilter, tvChart, tvData`: display labels
	`Button btnAddDataset, btnAddFilter`: buttons for opening dialogs `(AddDatasetDialogFragment, AddFilterDialogFragment)`
	`Button btnViewPatientList, btnViewHPIList`: buttons for opening activities `(PatientListActivity, HPIListActivity)`
	`RelativeLayout graphLayout`: chart holder
	`int schoolID`: school identifier of original dataset
	`String schoolName, date`: school name and date of original dataset
	`PieChart pieChart`: pie chart
	`BarChart barChart`: bar chart
	`ScatterChart scatterChart`: scatter chart
	`BubbleChart bubbleChart`: bubble chart
	`ArrayList<PatientRecord> allRecords`: records of all patients included in the dataset/s
	`ArrayList<PatientRecord> filteredRecords`: filtered records of patients after applying the specified filter/s
	`String[] xData`: labels in chart
	`int[] yData`: plotted values in chart
	`ArrayList<Dataset> datasets`: list of datasets available for adding
	`Spinner spRecordColumn, spChartType`: dropdown menu to choose what health test result and chart type to display
	`String recordColumn, chartType`: selected health test and chart type

METHODS	`refreshCharts()` – update changes to chart display `createCharts()` – create charts (pie, bar, scatter, bubble) `prepareChartData()` – prepare content of chart `createBubbleChart()` – create bubble chart with specifications `createScatterChart()` – create scatter chart with specifications `createBarChart()` – create bar chart with specifications `createPieChart()` – create pie chart with specifications `prepareFilterList()` – update recyclerview `addDatasetToList()` – update recyclerview `prepareRecord()` – get and set records from database `getSchoolID(): int` – get school identification from databse `getDatasetList()` – get dataset list from database

	`addDataSet()` – add dataset to chart depending on chart type `createEntries(): ArrayList<Entry>` – create entries to be added to chart `prepareBubbleChartData(ArrayList<Integer> colors)` – prepare entries to be added to bubble chart `prepareScatterChart(ArrayList<Integer> colors)` – prepare entries to be added to scatter chart `prepareBarChart(ArrayList<Integer> colors)` – prepare entries to be added to bar chart `preparePieChart(ArrayList<Integer> colors)` – prepare entries to be added to pie chart `filterRecordsByGender(String gender)` – filter records in `filteredRecords` by gender `fitlerRecordsByAge(String filterEquator, String filterValue)` – filter records in `filteredRecords` by age `getIndexByProperty(int value): int` – returns the index of the record with specified age

PATH	layout/HPIListActivity.java
CLASS	`HPIListActivity`
SUPERCLASS	`AppCompatActivity`
PROPERTIES	`String TAG`: identifier for the source of a log message `TextView tvTitle, tvName, tvDate`: display labels `RecyclerView rvHPIList`: list of HPIs `ArrayList<HPI> HPIList`: content of recyclerview `int schoolID`: school identifier of patient `int patientID`: identifier of patient
METHODS	`prepareHPIList()` – get and display HPI list from database

PATH	layout/MainActivity.java
CLASS	`MainActivity`
SUPERCLASS	`AppCompatActivity`
PROPERTIES	`String TAG`: identifier for the source of a log message `TextView tvTitle, tvUsername, tvPassword`: display labels `EditText etUsrname, etPassword`: user input holder `Button btnLogin, btnSignUp`: action buttons for either login or signup
METHODS	`checkDatabase()` – check if application database exists and create if not `saveUserToDatabase()` – insert user to database `checkUserFromDatabase()` – query user from database

PATH	layout/PatientListActivity.java
CLASS	`PatientListActivity`

SUPERCLASS	`AppCompatActivity`
PROPERTIES	`String TAG`: identifier for the source of a log message `int schoolID`: identifier for school of dataset `String date`: date of dataset `String recordColumn, columnValue`: strings needed for filtering patient list `RecyclerView rvPatientList`: list of patients `ArrayList<Patient> PatientList`: content of recyclerview `TextView tvSchoolName, tvName, tvGender, tvAge`: display labels
METHODS	`preparePatientList()` – get and display patient list from database

PATH	layout/HPIActivity.java
CLASS	`HPIActivity`
SUPERCLASS	`AppCompatActivity`
PROPERTIES	`String TAG`: identifier for the source of a log message `Spinner HPIDate`: date of available consultations `TextView tvTitle, tvHPIText, tvPatientName`: display content of HPIs `ArrayList<HPI> HPIs`: list of consultations of a patient `int HPIID`: identifier of currently displayed HPI `int patientID`: identifier of patient `Patient patient`: patient details
METHODS	`findHPI(int hpiID): HPI` – get HPI with the `hpiID` `prepareSpinner()` – prepare contents of spinner `perpareData()` – get HPIs from database

PATH	layoutViewImmunizationActivity.java
CLASS	`ViewImmunizationActivity`
SUPERCLASS	`AppCompatActivity`
PROPERTIES	`String TAG`: identifier for the source of a log message `Spinner spDate`: date of available immunization records `TextView tvTitle, tvName`: display details `ImageView ivImmunization`: picture of immunization record `ArrayList<Record> records`: list of records of a patient `int patientID`: identifier of patient `Patient patient`: patient details
METHODS	`prepareSpinner()` – prepare contents of spinner `prepareRecord()` – get records from database

PATH	layout/ViewPatientActivity.java
CLASS	`ViewPatientActivity`

SUPERCLASS	AppCompatActivity
PROPERTIES	String TAG: identifier for the source of a log message TextView tvName, tvBirthday, tvGender, tvDominantHand, tvRecordDate, tvBMI, tvPatientRemark, tvData, tvDate, tvHeight, tvWeight, tvVisualLeft, tvVisualRight, tvColorVision, tvHearingLeft, tvHearingRight, tvGrossMotor, tvFineMotorD, tvFineMotorND, tvFineMotorHold, tvRecordRemark: display content of patient details, record details, and labels ImageView ivPatient: picture of patient Button btnViewHPI, btnViewImmunization: buttons for next activity (ViewHPIActivity, ViewImmunizationActivity) ArrayList<Record> patientRecords: list of monitoring records of a patient ArrayList<Record> averageRecords: list of average results of monitoring records with the same age as the patient IdealValue idealValue: standard values indicating good health RelativeLayout graphLayout: holder of chart LineChart lineChart: line chart Spinner spRecordColumn: health tests available for viewing String recordColumn, chartType: string description of what health test and chart should be displayed int patientID: identifier of patient Patient patient: patient details
METHODS	displayRecord(Record record) – adjust raw data displayed prepareLineChartData() – create line data entries for chart getColumnValue(Record record) – get numerical value for recordColumn customizeChart(Chart chart) – customize chart appearance prepareLineChart() – customize line chart specific appearance createCharts() – create charts available addChartToView() – attach chart to layout getPatientData() – get patient details from database getPatientRecords() – get patient records from database getIdealValues(int age) – get standard values according to age getAverageRecords() – get average health test results of patients with the same age from database prepareRecordDateSpinner() – dropdown menu where dates of available records are displayed

C.5. Models

These model classes are created according to the content of the database and contains constant strings to refer to their respective table and columns (i.e. TABLE_NAME, C_<column_name>).

PATH	model/account/Dataset.java
CLASS	Dataset
PROPERTIES	String TAG: identifier for the source of a log message int datasetID: identifier for dataset int schoolID: identifier for school of dataset String schoolName: name of school of dataset

	`String dateCreated`: date of dataset `int status`: status of dataset (0 - not downloaded: 1 - downloaded)
METHODS	`Dataset((int datasetID, int schoolID, String schoolName, String dateCreated, int status)` – Constructor `printDataset()` – print dataset details in log

PATH	model/account/User.java
CLASS	`User`
PROPERTIES	`String TAG`: identifier for the source of a log message `int userID`: identifier for user `String username`: username of user `String password`: password of user `accessLevel`: access level of user (0 - no access; 1 - has access)
METHODS	`User(int userID, String username, String password, int accessLevel)` – Constructor

PATH	model/consultation/HPI.java
CLASS	`HPI`
PROPERTIES	`int hpiID`: identifier for HPI `int patientID`: identifier for patient `String dateCreated`: date when consultation was conducted `String hpiText`: content of HPI record
METHODS	`HPI(int hpiID, int patientID, String dateCreated, String hpiText)` – Constructor

PATH	model/consultation/Patient.java
CLASS	`Patient`
PROPERTIES	`int patientID`: identifier for patient `int schoolID`: identifier for school of patient `String firstName, lastName`: name of patient `String birthday`: birthday of patient `int gender`: sex of patient `int handedness`: dominant hand of patient (0 - right; 1 - left) `String remarksString`: remarks in text form `byte[] remarksAudio`: remarks in audio form
METHODS	`Patient(int pattientID, String firstName, String lastName, String birthday, int gender, int schoolID, int handedness, String remarksString, byte[] remarksAudio)` – Constructor `Patient(Parcel in)` – Constructor for parcel `readFromParcel(Parcel in)` – read contents of parcel

	`printPatient()` – print patient details in log `getAge(String date): int` – return age of patient at the given date

PATH	model/consultation/School.java
CLASS	`School`
PROPERTIES	`int schoolID`: identifier for school `String schoolName`: name of school `String municipality`: name of municipality
METHODS	`School(int schoolID, String schoolName, String municipality)` – Constructor

PATH	model/consultation/Record.java
CLASS	`Record`
PROPERTIES	`int recordID`: identifier for record `int patientID`: identifier for patient `String dateCreated`: date when monitoring activity was conducted `String gradeLevel`: educational level of patient at the date of activity `double height`: height of the patient `double weight`: weight of the patient `String visualAcuityLeft`: visual acuity of left eye `String visualAcuityRight`: visual acuity of right eye `String colorVision`: color vision `String hearingLeft`: hearing of left ear `String hearingRight`: hearing of right ear `int grossMotor`: gross motor `int fineMotorDominant`: fine motor of dominant hand `int fineMotorNDominant`: fine motor of non-dominant hand `int fineMotorHold`: fine motor of hold `byte[] vaccination`: picture of immunization record `byte[] patientPicture`: picture of patient `String remarksString`: remarks in text form `byte[] remarksAudio`: remarks in audio form
METHODS	`Record(int recordID, int patientID, String dateCreated, String gradeLevel, double height, double weight, String visualAcuityLeft, String visualAcuityRight, String colorVision, String hearingLeft, String hearingRight, int grossMotor, int fineMotorDominant, int fineMotorNDominant, int fineMotorHold, byte[] vaccination, byte[] patientPicture, String remarksString, byte[] remarksAudio)` – Constructor `Record(Parcel in)` – Constructor for parcel `readParcel(Parcel in)` – read parcel `printRecord()` – print record in log `getFineMotorString()` – get string result for fine motor test

PATH	model/consultation/IdealValue.java
CLASS	`IdealValue`
PROPERTIES	`int growthID`: identifier for ideal growth value `String recordColumn`: health test `int gender`: sex of patient `int year`: age of patient in years `int month`: age of patient in months after the age in years `float n3SD`: standard value -3 standard deviation (SD) from median `float n2SD`: standard value -2 SD from median `float n1SD`: standard value -1 SD from median `float median`: standard value `float p1SD`: standard value 1 SD from median `float p2SD`: standard value 2 SD from median `float p3SD`: standard value 3 SD from median
METHODS	`IdealValue(int growthID, String recordColumn, int gender, int year, int month, float n3sd, float n2sd, float n1sd, float median, float p1sd, float p2sd, float p3sd)` – Constructor `printIdealValue()` - print ideal value contents to log

C.6. Others

These are classes which do not belong to other categories but are used for calculation and such.

PATH	model/account/PasswordEncrypter.java
CLASS	`PasswordEncrypter`
PROPERTIES	`SecretKeySpec sks`: key for encryption
METHODS	`PasswordEncrypter()` – Constructor `encryptPassword(String password): byte[]` – encrypt password and return as a byte array `decodePassword(byte[] encodedPassword): String` – decode encoded password and return as string