*Design document*

Analysis app for slum development

For



By



Fifth Floor, Shree Niketan,

Sheelavihar Colony,

Off Karve Road,

Pune – 411 038.

India.

Tel.: +91-20-25460736/25425910

Fax: +91-20-25451908

Email: arun@soft-corner.com

Ref: SC/SA/1516/2

April 4, 2016

# Contents

[Contents 2](#_Toc447562976)

[Introduction 3](#_Toc447562977)

[Key Features 4](#_Toc447562978)

[Database design 10](#_Toc447562979)

[Screen Layouts 17](#_Toc447562980)

[KoboCat APIs 18](#_Toc447562981)

# Introduction

This document lists down the database design, some key screen layouts (wireframes), some explanation about how particular features will be implemented and the API of Kobo Toolbox that would be used.

# Key Features

Shelter Associates website will have the following main links…

1. Surveys
2. Analysis
3. Reporting
4. Sponsors
5. Administration

**Survey Creation**

1. Surveys will be created using KOBO toolbox. There will be 4 types of surveys…
2. Rapid Appraisal Survey (RA)– this will be one survey record per slum – to gather slum info
3. Rapid Household Survey (RHS)– this will have multiple records per slum – gathers info per house in a slum
4. Household Member Survey (SE Socio Economic survey) – this will have info on each member of each house in a slum. This is not mandatory and may not be taken for a slum
5. Family Factsheet Survey (FF) – this is taken after intervention - also one every house where toilet is built.

Each Survey type for each city will have a different URL

Eg. //ShelterSurvey\_Slums\_Pune could be URL for Survey type A for Pune

//ShelterSurvey\_Slums\_Pune\_Household could be URL for Survey type B for Pune

//ShelterSurvey\_Slums\_Vashi could be URL for Survey type A for Vashi

1. For one city, the Slum Survey (Survey A) will be defined and deployed.
   * This survey can be copied for another City and modifications (if any) can be done for the new city.
   * Separate URL will be given for a survey of another city.
   * Same logic will apply for Survey B and Survey C.
2. Each of these surveys will have one first question that will be a drop-down that shows a list of all slums in the city.
   * This survey question will be created using the list of slums defined in the table ‘Slum’.
   * All slums in the city would be added first in this list before starting the first survey in that city.
   * If at a later date, a slum is added, then this will directly get added as an option in the options list of the dropdown field (slum) in the survey defined using Kobo Toolbox.
   * The List of slum will contain slum names – which will be visible to the user and Id for that list will be the Shelter\_Slum\_Code. (refer Slums table)
3. Whenever a new survey gets deployed, there will be an alert given in the web application site for the Shelter user to create an entry of this new survey in the ‘Survey’ Table. This ensures that we have corresponding metadata in our application to facilitate analysis of survey etc. This alert can be achieved using a cron job.
4. Show a list of Ongoing Surveys, Past Surveys and Surveys yet to begin with some statistical information like Survey Status (Defined, Active, Closed), Last updated on, Count of records.
5. Analysis for any survey should not be enabled till a threshold number of responses are there (which is a field in the survey table).

**Survey Analysis**

1. A separate menu for Analysis will be available. Analysis will be possible at
2. City Level
3. Administrative ward level
4. Electoral ward level
5. Slum level

City level analysis can be done only on Survey Type A. Other analyses will mostly be done on Survey type B and D.

1. Data cleaning: Some sanity check on the data has to be performed and exceptions need to be highlighted and corrected / removed. Eg. One slum survey contains same House numbers appearing in multiple records etc. Need to define some queries that will give exception records.
2. All responses of the survey will be displayed, filters can be applied and data can be sorted by columns. The results can be exported to excel
3. Query creation – Give a UI to allow users to create and save the queries for analysing the Survey data. Graphs, Cross tabulation, Frequencies etc. can be computed on filtered or unfiltered data
4. Retrieve saved Queries and run analysis and show the results on screen
5. Export the result in excel / pdf along with some description

**More on Queries and Filters**

1. Some Queries and Filters can be created and saved in the system by Shelter Admin / Power users. These saves filters/Queries can be published so that they are available to different users based on their Access Rights.

So Filters / queries can have a status like

1. Public – available to all users
2. Request – some user could ask for specific information and Shelter user can create and make that available to specific users
3. Private – created by individual users and used only by him/her

However, when a query is created care has to be taken that the result of that query will give data that is based on that users access rights (of city/slum)

The filter itself will be de-linked from the report that is to be seen. i.e. a filter will only decide the dataset. The columns in a report or a cross tabulation or a graph will be called as a report. The report will use the data from the data that has been filtered using the filter.

For example, a filter could be set to filter from selected city, all slums where there are more than 2 Community toilet blocks and those who are ready to opt for a toilet in their home. The data that is to be viewed will be defined in a report - which may be a tabular report like ‘Slum name, household number, number of persons in the family, family income’ OR it could be a slum level graph that shows toilet seat ratios for male and female in such slums.

More detailing of this needs to be done.

**Reports**

Different Reports have to be generated at different levels

1. Survey result view.
2. Family Factsheet – One report per family after construction of toilet. This is in PDF format. – This is based on Survey B and Survey D. UI to add photographs and maps.
3. Drainage Report – pre and post intervention – using the Survey A. Map to be added.
4. Vulnerability Report – Based on Survey A – Excel output – City based
5. Rapid factsheet (Created after survey A) – Slum level information
6. RA Report (4 to 5 pages, some info from RA, some from RHS)
7. Socio economic report
8. Impact assessment report
9. Presentations (ward level presentation, electoral ward level, slum level)

**Sponsors**

Many corporates, individuals sponsor / fund these projects. Funding could be done for conducting surveys, building toilets or both. Such funding entities should be given access to Shelter website to login and get reports of their fund allocations.

1. Capture data of the Sponsors – Company Name, Address, Contact persons
2. Create users with a role of Sponsor – could be multiple users of one company
3. Give a login access to these users
4. Admin user can Associate Survey to Sponsor
5. Admin user can Associate Toilets to Sponsor – could be from multiple slums and multiple cities.
6. Allow Sponsor to View and export Family Factsheet report with photos, maps of their sponsored projects. Should be possible to get separate reports by financial year.

Note: Commercials related to the Sponsors funds and their allocations to the toilets etc. is not in the scope of this project.

**Administration**

1. **Users, Roles and Access rights**

User Master – Name, eMailId, Contact nos, UserRole

There are eight user roles:

* Admin (Shelter)
* Power User (Shelter)
* Survey User
* Govt Of India User
* State Govt User
* Urban Local body user
* Other NGO
* Funder

Guest users are people who come to Shelter website and can view information without logging in

It is not possible to use the role-access mechanism provided by Django – as this is table based. We need to create roles based on data values in particular columns in particular records.

Role Master: Admin will define different access settings for each role in the system. This basically tells us, what is the granularity of access rights that is possible for each type of user.

* City Access – (All or specific)
* Slum Access – (All or Specific)
* Can access published queries
* Can create own queries
* Can Request Specific reports
* Can access Slum Fact Sheet
* Can access Family Factsheet
* Can Access RA and RHS data
* Can view Vulnerability report
* Can view Drainage report

Access Areas: This table will store which User can have access to which City and which slum. So multiple records for one user will be stored.

Every user will be assigned a role and can be given access to Specific cities and slums.

This design becomes scalable to define more roles and also add access rights in future.

1. **Creating Master data**
   1. City
   2. Administrative Ward
   3. Electoral Ward
   4. Slum
   5. WardOfficeContacts
   6. Elected Representatives
   7. Code Master
2. **Housekeeping**

* Purging old data
* Reset Password
* Define Password Policy

**One time activity**

Once the development is complete, it would be necessary to plan and design a process for migrating all the old surveys and their data into the current database.

# Database design

Kobo Toolbox creates tables for the survey in both MongoDB as well as ProgressDB. We have chosen to go with the ProgressDB as the querying for analysis would be simpler with a relational DB.

For the survey, Kobo Toolbox creates its own tables – which we would normally never be required to access, as we have the API to get and insert data. The tables below are the additional ones we need for our own masters, filters, reports and analysis.

The colour codes used are…

|  |  |
| --- | --- |
|  | Primary key |
|  | Foreign key |

**Table Name: Survey**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Name | Varchar(50) | Unique |  |
| Description | Varchar(200) | Nullable | Information about Survey |
| Type | int |  | Slum level, household level, household member level etc. |
| AnalysisThreshold | int |  | Number of responses before which analysis is not enabled |
| KobotoolSurveyId | Varchar(20) |  | Kobocat survey form Id |
| KobotoolSurveyUrl | Varchar(512) |  | Kobocat survey form URL |
| City\_id | Int | FK | City |

**Table: City**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Name | Varchar(50) | Unique |  |
| Shape | Text |  | City Lat Long |
| StateCode | Varchar(5) | Unique | State Code |
| DistrictCode | Varchar(5) | Unique | District Code |
| CityCode | Varchar(5) |  | City Code |
| CreatedBy | int |  |  |
| CreatedOn | Date |  |  |

**Table: AdministrativeWard**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Name | Varchar(512) | Unique |  |
| Shape | Text |  | AdministrativeWard Lat Long polygon |
| WardNo | Varchar(10) |  |  |
| Description | Text |  |  |
| OfficeAddress | Text |  |  |
| City\_id | Int | FK | City |

**Table: ElectoralWard**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Name | Varchar(512) | Unique |  |
| Shape | Text |  | Electoralward Lat Long polygon |
| WardNo | Varchar(10) |  |  |
| WardCode | Varchar(10) |  |  |
| Description | Text |  |  |
| AdministrativeWard\_id | Int | FK | AdministrativeWard |

**Table: Slum**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Name | Varchar(512) |  |  |
| Shape | Text |  | Slum Lat Long polygon |
| Description | Text |  |  |
| ElectoralWard\_id | Int | FK | Electoralward |
| ShelterSlumCode | Varchar(11) | Unique | Unique Slum Code, created by shelter |

**Table: WardOfficeContacts**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | int | PK |  |
| Name | Varchar(200) |  |  |
| Title | Varchar(25) |  |  |
| Telephone | Varchar(50) | Nullable |  |
| AdministrativeWard\_id | int | FK | Administrative Ward |

**Table: ElectedRepresentative**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Name | Varchar(200) |  |  |
| Telnos | Varchar(50) |  |  |
| Address | Text |  |  |
| Postcode | Varchar(10) |  |  |
| AdditionalInfo | Text |  |  |
| Party | Varchar(100) |  |  |
| ElectoralWard\_id | Int | FK | Electoralward |

**Table: Sponsor**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Organization | Varchar(200) |  |  |
| Address | Text |  |  |
| PhoneNumber | Varchar(50) |  |  |
| Description | Text |  | Addition Information of sponsor |
| Image | Varchar(500) |  | URL of an image (could be a logo of the sponsor) |

**Table: SponsorProject**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Name | Varchar(512) |  |  |
| Type | int |  | Selection for intervention or collection |
| Sponsor\_id | int | FK | Sponsor |
| CreatedBy | int |  |  |
| CreatedOn | Date |  |  |

**Table: SponsorProjectMetadata**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | int | PK |  |
| HouseholdCode | int |  | Household code created by shelter |
| Slum\_id | int | FK | Slum |
| SponsorProject\_id | int | FK | Sponsor Project |

**Table: SponsorUser**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Sponsor\_id | Int | FK | Sponsor |
| User\_id | Int | FK | auth\_user |

**Table: FilterMaster**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Name | Varchar(512) |  |  |
| IsDeployed | bit |  | Flag for filter deployed or not |
| VisibleTo | int |  | 1(All), 2(individual), 3(user type) |
| CreatedBy | int |  |  |
| CreatedOn | Date |  |  |

**Table: Filter**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | int | PK |  |
| Query | text |  |  |
| FilterMaster\_id | int | FK | Filter Master |

**Table: FilterMasterMetadata**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | int | PK |  |
| User\_id | int | FK | auth\_user |
| UserType | int | FK | auth\_group |
| Filter\_id | int | FK | Filter\_Master |

**Table: RoleMaster**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | int | PK |  |
| RoleName | varchar(512) |  |  |
| City | int |  | 0, 1(All), 2(allow selection) |
| Slum | int |  | 0,1(All),2(allow selection) |
| KML | bit |  | Can create KML |
| DynamicQuery | bit |  | Can create Dynamic Query |
| PredefinedQuery | bit |  | Can create Predefined Query |
| CanRequest | bit |  | Flag for download reports |
| Users | bit |  | Can create User |
| CreateSaveQuery | bit |  | Can create CreateSaveQuery |
| DeploySurvey | bit |  | Can DeployeSurvey |
| UploadImages | bit |  | Can UploadImage |
| PrepareReports | bit |  | Can PrepareReports |

**Table: UserRoleMaster**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | int | PK |  |
| User\_id | int | FK | auth\_user |
| Role\_id | int | FK | RoleMaster |
| City\_id | int | FK | City |
| Slum\_id | int | FK | Slum |

**Table: ShapeCode**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Code | Varchar(25) | Unique | line, point |
| Description | Varchar(10) |  |  |

**Table: DrawableComponent**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Name | Varchar(50) | Unique |  |
| Color | Varchar(10) |  |  |
| ExtraInformation | Text |  |  |
| Marker\_icon | Varchar(500) |  | URL of an image |
| ShapeCode\_id | Int | FK | ShapecCode |

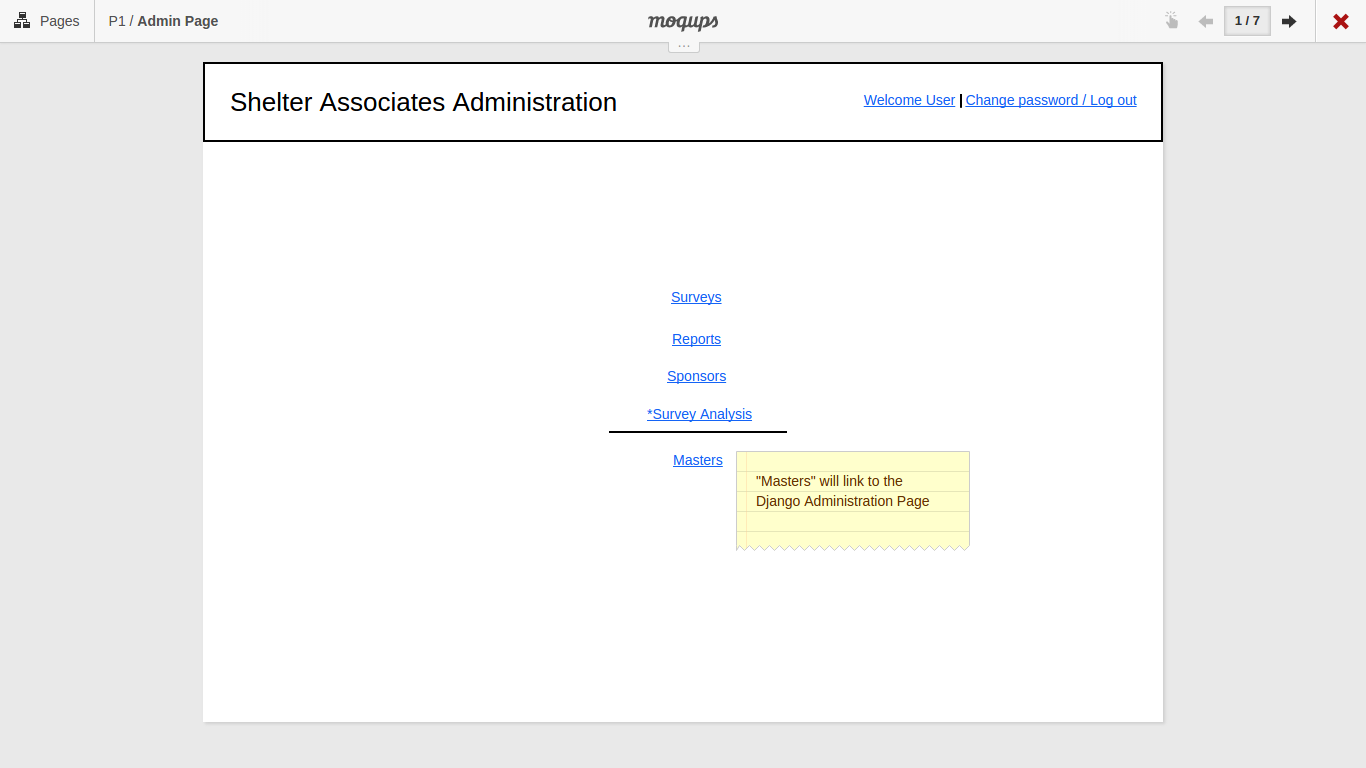
**Table: PlottedShape**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** |  | **Comments** |
| Id | Int | PK |  |
| Slum | Varchar(10) | Unique |  |
| Name | Varchar(512) |  |  |
| Lat\_long | Varchar(50) |  | Lat Long of Plotted Shape |
| DrawableComponent\_id | Varchar(10) | FK | DrawableComponent |
| CreatedBy | int |  |  |
| CreatedOn | Date |  |  |

# Some Screen Layouts

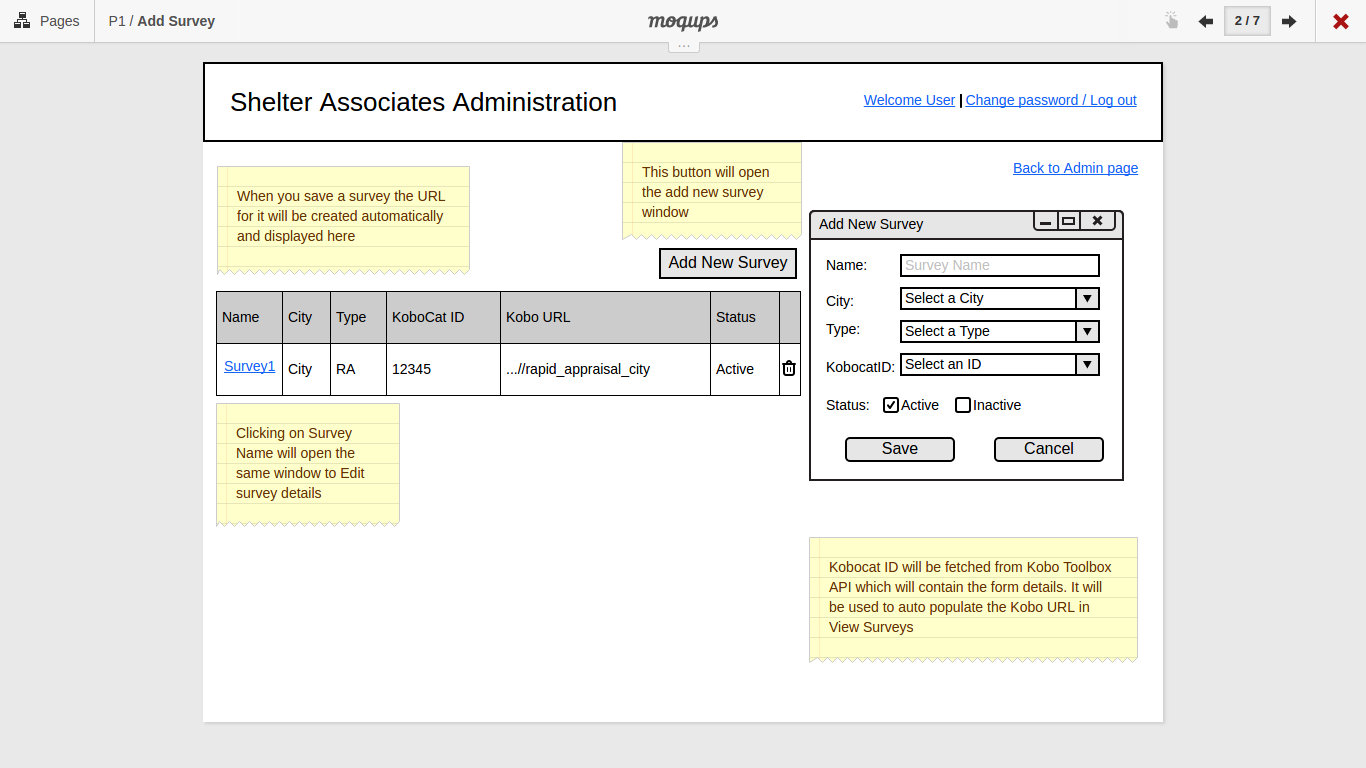
The screen layouts given below are for some key pages and are not yet complete. For example, the landing pages for each have to be designed – as those need further discussions with Shelter Staff. Also, for the master screens, the layout would be more or less as provided by Django. The ones that are different are given below. These are only wireframes and have to be refined further.

**Administration page**



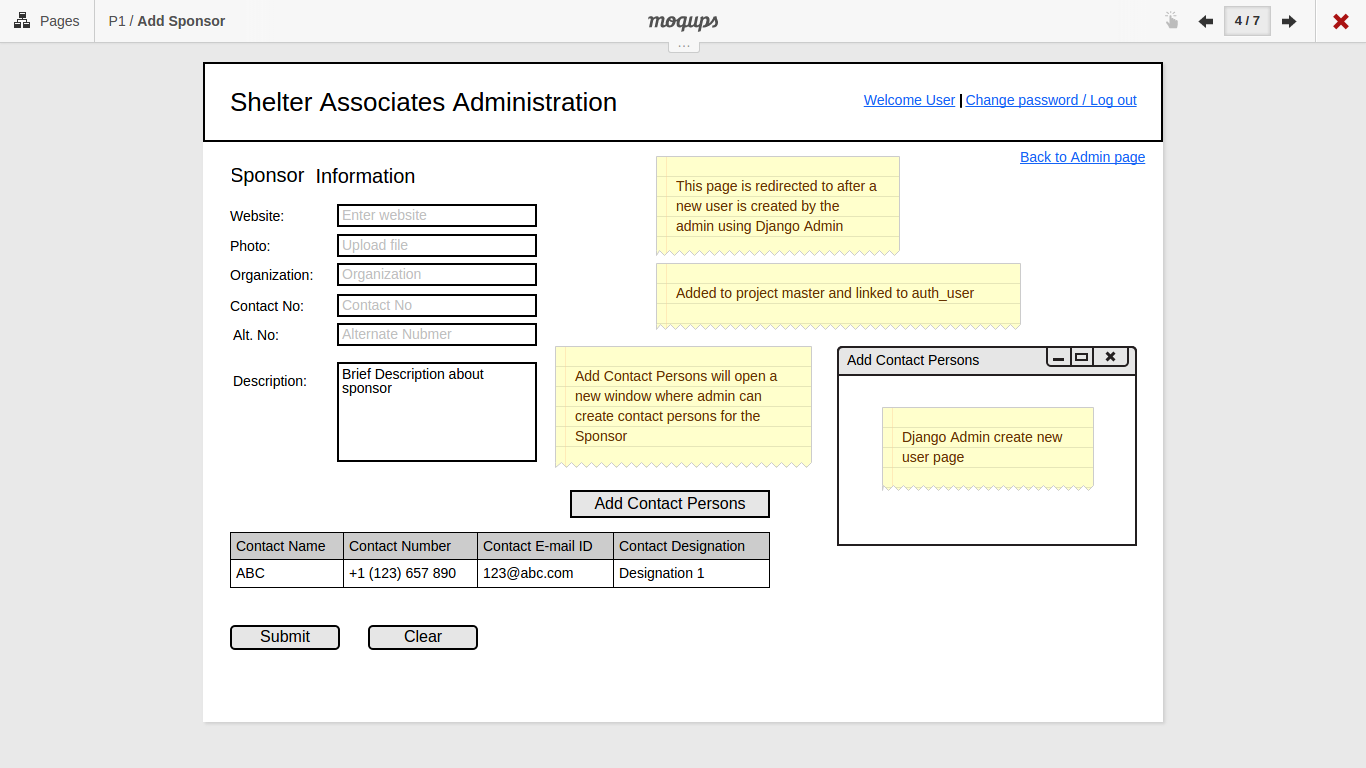
The above is a place holder and this would change later after further discussions with Shelter Staff.

Adding a Survey.

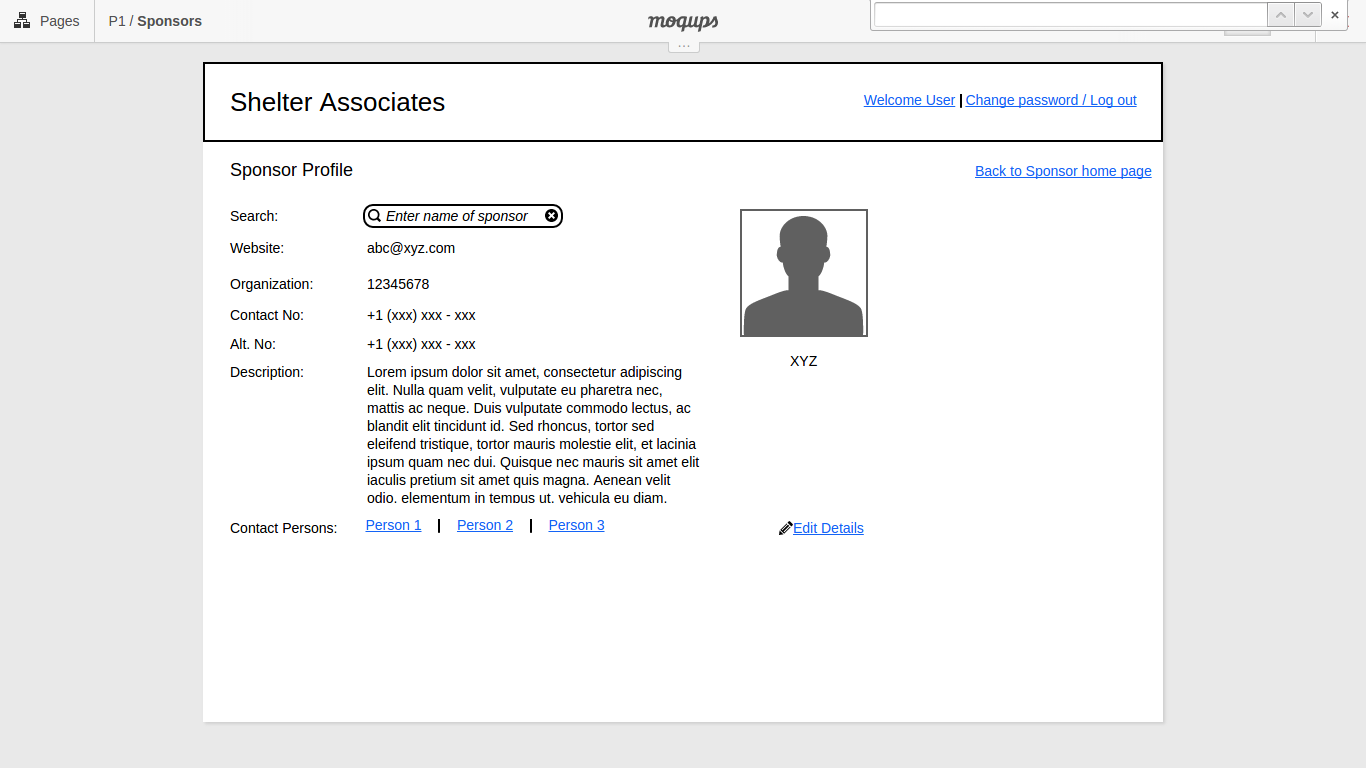


The above screen shows how a survey would be added and mapped with the Kobo Toolbox survey.

Adding Sponsor information

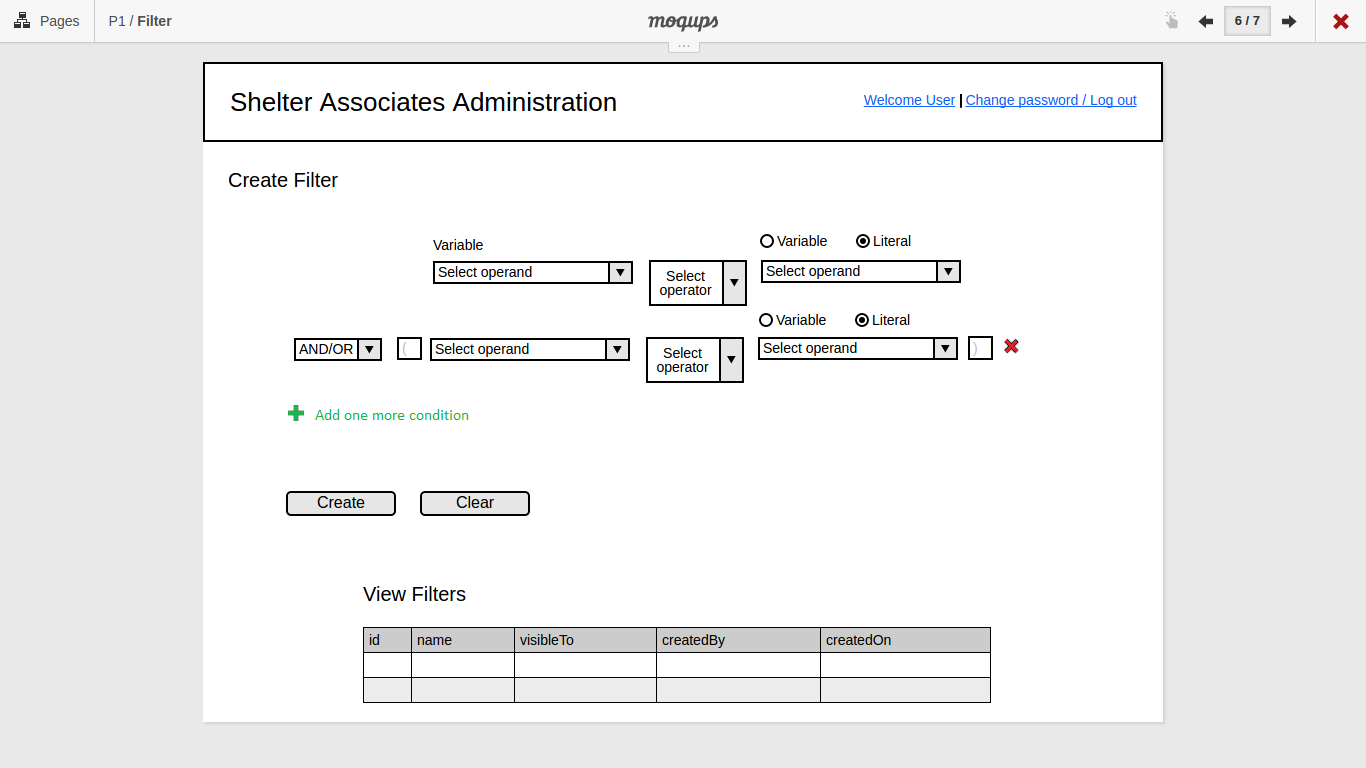


Information about the sponsor will be captured here.



Further screens will be available to allocate exact households to a sponsor’s project – where the relationship between the household in a slum and a sponsor is created. Once this is there, the sponsor will be able to directly see all the households where the intervention has been done – and check if the work is completed. This screen is yet to be designed.

Creating filters



The above loosely shows how the filter will be designed. Further refinement is needed in the above screen.

# KoboCat APIs

The APIs that are needed to exchange data between Kobo Toolbox and the Shelter App are listed below. Most of these could be needed – however not all of these may be needed. One can say that, what are needed would definitely be in the list.

| **API** | **Request Method** | **Sub URL and Input Parameters, if required** | **Output Keys** |
| --- | --- | --- | --- |
| **CHARTS** | | | |
| List of chart chart endpoints accessible to registered user | GET | /api/v1/charts | id, id\_string, url |
| Get a list of chart field endpoints for a specific form or dataset. | GET | /api/v1/charts/{formid} | id, id\_string, url, fields(form fields) |
| Get a chart for a specific field in a form | GET | /api/v1/charts/{formid}.{format}?field\_name=field\_name | html format response is a html, javascript and css to the chart |
|  |  | format can be json or html | json format response is the JSON data that can be passed to a charting library |
| Get a chart data for all fields in a form | GET | /api/v1/charts/{formid}?fields=all | json format response is the JSON data for each field that can be passed to a charting library |
| **DATA** |  |  |  |
| JSON List of data end points | GET | /api/v1/data | id, id\_string, url, title, description |
| Download data in csv format | GET | /api/v1/data.csv | CSV Data File |
| JSON List of data end points filter by owner | GET | /api/v1/data?owner=owner\_username | Lists the data endpoints accessible to requesting user, for the specified owner as a query parameter. |
|  |  | owner - username of the owner(user/organization) of the data point |  |
| **FORMS** |  |  |  |
| Submitted data for a specific form | GET | /api/v1/data/{pk} | id, bamboo\_dataset\_id, deleted\_at,expense\_type,xform\_id\_string,geolocation,end,start,status,uuid,imei, submission time, attachments |
|  |  | pk - the form unique identifier(Primary Key) |  |
| Query submitted data of a specific form using Tags | GET | /api/v1/data?tags=tag1,tag2 | Provides a list of json submitted data for a specific form matching specific tags, tags should be a comma separated list of tags |
|  |  |  | e.g https://example.com/api/v1/data/22845?tags=monthly |
| Tag a submission data point | POST | /api/v1/data/{pk}/{dataid}/labels | {"tags": "tag1, tag2"} |
| Delete a specific tag from a submission | DELETE | /api/v1/data/{pk}/{dataid}/labels/tag\_name | HTTP 200 OK |
| List of public data endpoints | GET | /api/v1/data/public | id, id\_string, url,title,description |
| Enketo edit link for a submission instance | GET | /api/v1/data/{pk}/{dataid}/enketo | {"url": "https://hmh2a.enketo.formhub.org"} |
| Delete a specific submission instance | DELETE | /api/v1/data/{pk}/{dataid} | HTTP 204 No Content |
| Single data submission for a given form | GET | /api/v1/data/{pk}/{dataid} | id, bamboo\_dataset\_id,deleted\_at,expense\_type,xform\_id\_string,geolocation,end,start,status,uuid,imei, submission time, attachments |
| **META DATA LIST** |  |  |  |
| Get list of metadata | GET | /api/v1/metadata | Returns a list of metadata accross all forms requesting user has access to. |
| Get list of metadata for a specific form | GET | /api/v1/metdata?xform=formid |  |
| Get a specific metadata | GET | /api/v1/metadata/{pk} |  |
| Add metadata or media file to a form | POST | /api/v1/metadata |  |
| Delete Metadata | DELETE | /api/v1/metadata/{pk} |  |
| **MEDIA** |  |  |  |
| Lists attachments of all xforms | GET | /api/v1/media/ |  |
| Retrieve details of an attachment | GET | api/v1/media/{pk} |  |
| Retrieve an attachment file | GET | /api/v1/media/{pk}.{format} |  |
| Lists attachments of a specific xform | GET | /api/v1/media/?xform={xform} |  |
| Lists attachments of a specific instance | GET | /api/v1/media?instance={instance} |  |
| Retrieve image link of an attachment | GET | api/v1/media/{pk} | HTTP 200 OK |
| **PROJECT LIST** |  |  |  |
| Register a new Project | POST | /api/v1/projects |  |
| List of Projects | GET | /api/v1/projects |  |
| List of Projects filter by owner/organization | GET | /api/v1/projects?owner=owner\_username |  |
| Retrieve Project Information | GET | Retrieve Project Information |  |
| Update Project Information | PUT | /api/v1/projects/{pk} |  |
| Update Project Information | PATCH | /api/v1/projects/{pk} |  |
| Share a project with a specific user | POST | /api/v1/projects/{pk}/share | HTTP 200 OK |
| Send an email to users on project share | POST | /api/v1/projects/{pk}/share | HTTP 204 NO CONTENT |
| Remove a user from a project | POST | /api/vi/projects/1/share | HTTP 204 NO CONTENT |
| Assign a form to a project | POST | /api/v1/projects/{pk}/forms |  |
| Upload XLSForm to a project | POST | /api/v1/projects/{pk}/forms |  |
| Get forms for a project | GET | /api/v1/projects/{pk}/forms |  |
| Get list of projects with specific tag(s) | GET | /api/v1/projects?tags=tag1,tag2 |  |
| Get list of Tags for a specific Project | GET | /api/v1/project/{pk}/labels | Response HTTP 200 OK |
| Tag a Project | POST | /api/v1/projects/{pk}/labels |  |
| Remove a tag from a Project | DELETE | /api/v1/projects/{pk}/labels/tag\_name | HTTP 200 OK |
| Add a star to a project | POST | /api/v1/projects/{pk}/star |  |
| Remove a star to a project | DELETE | /api/v1/projects/{pk}/star |  |
| Get user profiles that have starred a project | GET | /api/v1/projects/{pk}/star | HTTP 200 OK |
|  | GET | /api/v1/projects | HTTP 200 OK |
| **USERS AND ORGANIZATIONS** |  |  |  |
| Register a new Organization | POST | /api/v1/orgs | org,name,email,city,country |
| List of Organizations | GET | /api/v1/orgs | Url,org,name,email,city,country,website,twitter,gravatar,require\_auth,user,creator |
| Retrieve Organization Profile Information | GET | /api/v1/orgs/{username} | Url,org,name,email,city,country,website,twitter,gravatar,require\_auth,user,creator |
| List Organization members | GET | /api/v1/orgs/{username}/members | ["member1", "member2"] |
| Add a user to an organization | POST | /api/v1/orgs/{username}/members | ["member1"] |
|  |  | To add a user to an organization requires a JSON payload of {"username": "member1"}. |  |
| Change the role of a user in an organization | PUT | /api/v1/orgs/{username}/members | ["member1"] |
|  |  | To change the role of a user in an organization pass the username and role {"username": "member1", "role": "owner|manager|editor|dataentry|readonly"}. |  |
| Remove a user from an organization | DELETE | /api/v1/orgs/{username}/members | [] |
|  |  | To remove a user from an organization requires a JSON payload of {"username": "member1"}. |  |
| Register a new User | POST | /api/v1/profiles | username,name,email,city,country,... |
| List User Profiles | GET | /api/v1/profiles | Url,organization,email,city,country,website,twitter,gravatar,require\_auth,user,user |
| Retrieve User Profile Information | GET | /api/v1/profiles/{username} OR | Url,organization,email,city,country,website,twitter,gravatar,require\_auth,user,user |
|  |  | /api/v1/profiles/{pk} |  |
| Partial updates of User Profile Information | PATCH | /api/v1/profiles/{username} | Selectively update user profile information |
|  |  | Payload required is for properties that are to be changed in JSON, for example, {"country": "IN"} will set the country to IN. |  |
| Change authenticated user's password | POST | POST -d current\_password=password1 -d new\_password=password2 /api/v1/profile/demouser/change\_password | HTTP 200 OK |
| Get List of Teams | GET | /api/v1/teams | Url,name,organization,email, Url,name,organization,email,Url,name,organization,email,... |
| Get Team Info for a specific team. | GET | /api/v1/teams/{pk} | Url,name,organization,email |
| List members of a team | GET | /api/v1/teams/{pk}/members | ["member1"] |
| Add a user to a team | POST | {"username": "someusername"} to /api/v1/teams/<pk>/members | ["someusername"] |
| Retrieve profile | GET | /api/v1/user | api\_token,city,country,gravatar,name,organization,require\_auth,twitter,url,user,username,website |
| Get projects that the authenticating user has starred | GET | /api/v1/user/{username}/starred | [“Project1”] |
| Request password reset | POST | /api/v1/user/reset | HTTP 204 OK |
| Reset user password | POST | uid, token and new\_password are expected in the POST payload. Minimum password length is 4 characters | HTTP 204 OK |
| List Users | GET | /api/v1/users | username,first\_name,last\_name,username,first\_name,last\_name,... |
| Retrieve a specific user info | GET | /api/v1/users/{username} | Id,username,first\_name,last\_name,url... |