# Dandelion data collection platform

System specification

## Architectural overview

SRUC sever

Python Flask app

REST API

Database

File system

files

├─ 123

│ ├─ 66

│ │ ├─ file1

│ │ ├─ file2

│ │ └─ file3

React app

Local authority network

School WiFi network

GrowCube

Lilygo TTGO T5

Sensors

Battery

Internet



## User roles

|  |  |
| --- | --- |
| Role | Description |
| Sysadmin | Superuser with access to all parts of the system for development, maintenance, management and problem resolution |
| Superuser | Designated staff member(s) within a school with management responsibility for the school’s users and content |
| Project leader | School user with delegated responsibility for a specific project. They can manage experiments including assigning experiment participants. |
| Experiment participant | School user participating in a specific experiment. They are allowed to upload experimental observations and create reports. |
| School user | Staff or student user without management responsibility. The system does not differentiate between staff and students. A user may be part of a project team with additional privilege to update the project data. |
| Public | Unauthenticated user visiting the public site to view content and data |

## Database structure conventions

The following column definitions will be used in all tables where required:

|  |  |  |
| --- | --- | --- |
| Column | Datatype | Description |
| id | Integer | Synthetic primary key |
| description | Varchar(120) | Short description of the entity represented by the record |
| created\_date | Datetime | Record creation timestamp |
| updated\_date | Datetime | Timestamp of last update to this record |
| created\_by | Integer | Foreign key to the users table |
| status | Char(2) | Status code |

Columns that represent Boolean flags have names starting with is\_ (e.g. is\_superuser).

Foreign key columns names include the name of the related table (e.g. school\_id) unless the name is context-specific (e.g. created\_by).

Columns that represent files held in the file system have names ending in \_link (e.g. project\_image\_link).

Columns containing large bodies of text (>255 characters) have names ending in \_text (e.g. project\_text).

## Auditing

The auditing function tracks changes made to the records in certain tables in the database. All updates are recorded so that the AUDIT table contains a complete trace of the record’s history. This functionality is handled by the back-end and is exposed to the front-end via the API schemas.

The AUDIT table records the following data:

|  |  |
| --- | --- |
| Column | Description |
| id | Unique identifier |
| user\_id | FK to the USERS table: identifies the user making the change |
| audit\_date | Timestamp of the change event |
| table\_name | Name of the table being updated |
| primary\_key\_value | PK of the record being changed (assumed to be called **id**) |

Note that to retrieve the history of a record, the AUDIT table needs to searched by explicitly using the table name and primary key value. This method simplifies the database structure. An index is required on those two columns.

The details of the actual changes are stored in the AUDIT\_DETAIL table which has the following columns:

|  |  |
| --- | --- |
| Column | Description |
| id | Unique identifier |
| audit\_id | FK to the AUDIT table |
| column\_name | Name of the column being updated |
| old\_value | Previous value of the column. This field is NULL for the first audit record that marks the row’s creation. For later updates, the new value is stored in the row itself. |

The following tables are audited:

AUTHORITY

CONDITION

EXPERIMENT

HYPOTHESIS

NODE

NODE\_SENSOR

PROJECT

PROJECT\_LEADER

PROJECT\_PARTNER

REPORT

SCHOOL

TAG\_REFERENCE

USERS

VARIABLE

## Projects

A project may be specific to one school, or may be shared by many schools. Participation is modelled by the bridge table, PROJECT\_PARTNER. Where a project is specific to one school, there will be a single record in this table.

Shared projects may include all schools or a subset. The structure supports either arrangement, and the setup of shared projects is handled by application code rather than being part of the database structure.

It is assumed that for all shared projects there will be one school designated as lead partner. This is to provide accountability. It is possible for the lead partner to change.

For all projects, there will be a specific person in each participating school who is designated as the project leader. This means that one project may have several project leaders, but only one per school. This needs to be enforced by application code.

## Experiments

Experiments are carried out by one or more users in a specific school. They are created by a project leader who assigns participants from among school users. Experiment participants may then update the experiment details and add observations and reports.

For shared projects, it is likely that participating schools will want to carry out related experiments. This is indicated by the PARTNER\_ID column in the EXPERIMENT table which is a foreign key to PROJECT\_PARTNER. Once an experiment is defined, it should be possible to duplicate it for other partner schools. If the Boolean column IS\_SYNCHRONISED is set to TRUE, application code should ensure that only the project leader from the lead school is allowed to update the experiment details. This includes data in the tables EXPERIMENT, HYPOTHESIS, CONDITION, VARIABLE, TAG. Experiment participants in all schools are still allowed to record data and create reports.

Experiments may have defined hypotheses but these are optional. Multiple conditions are also optional, but every experiment must have at least one condition. Where an experiment only has one condition, it should be suppressed in the UI.

## Variables

A variable is defined for an experimental condition. Where the experiment has multiple conditions, the same variables will probably be defined for each one but there may be some variation. Once the variables for one condition have been defined, application code should allow those variables to be duplicated for other conditions.

A variable may be measured manually or by a sensor in the IoT node. This is indicated by the FK column VARIABLE.NODE\_SENSOR\_ID which is NULL for manually-measured variables.

The column VARIABLE.VARIABLE\_ROLE indicates the role of the variable in the experimental design. Valid values are *independent*, *dependent*, *control* or NULL.

## Observations

Data that is measured by IoT node sensors is recorded in csv files in the file system. Therefore, no records are required in the OBSERVATION table.

Manually-recorded data points are recorded as rows in the OBSERVATION table. An observation may consist of a numerical value, an image or both.

## Charts