Vanessa: First Look And partial walk-through

2018.04.27.et; last update: 2018.05.07

Introduction

As of this writing, Vanessa is very new software. There are bugs and feature omissions. This is introductory guide to assist in evaluating the software.

URLs

There are two URLs. The *production* URL provides a recent stable snapshot of the software. In contrast, the *development* URL contains the latest development code which may be unstable.

The current *production* URL is: https://icamlinux.surg.med.umich.edu/v0/
The current *development* URL is: https://icamlinux.surg.med.umich.edu/v3/

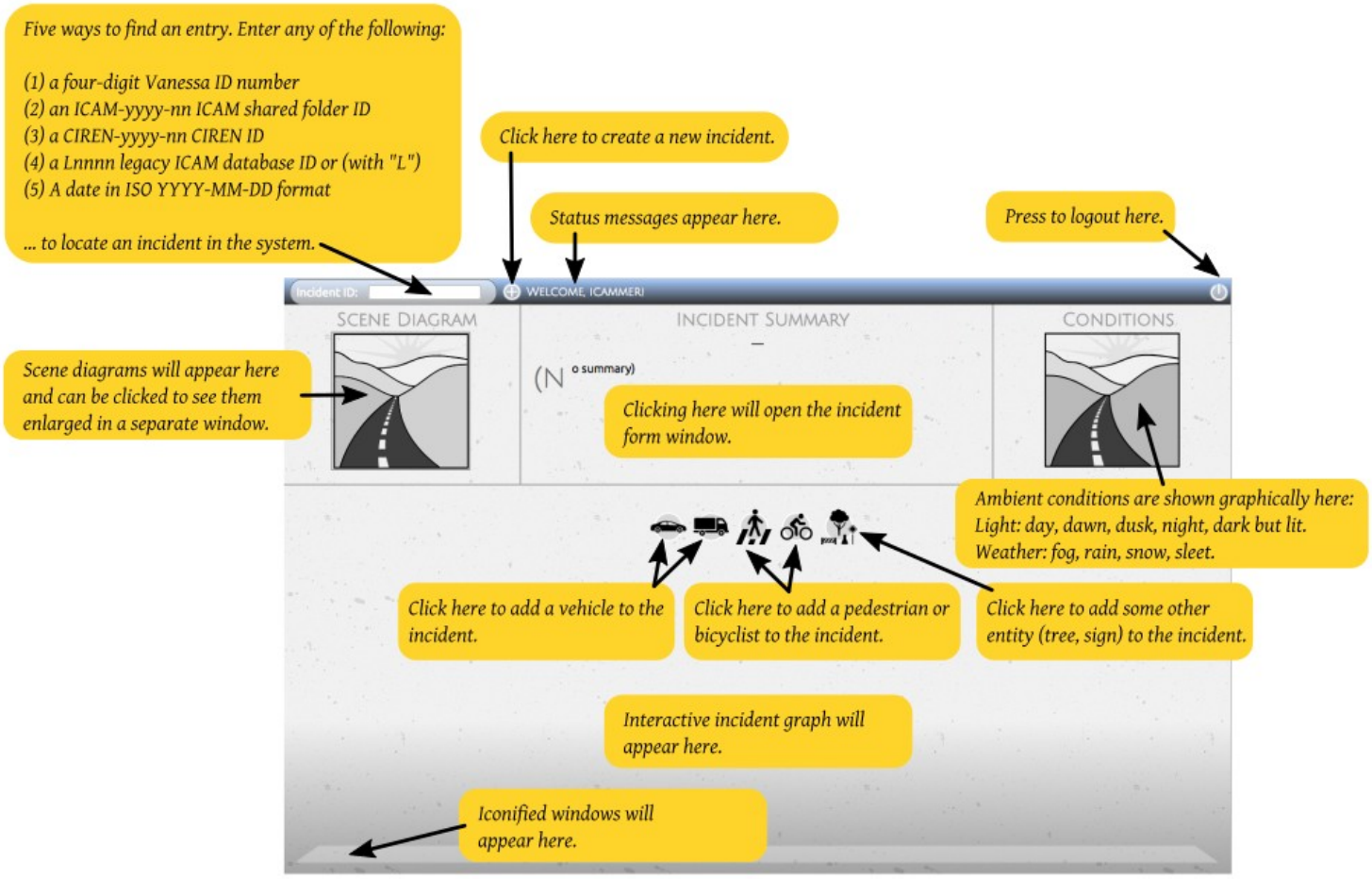
Logging In

The login screen looks like this:

Ask Ed for credentials to log in, then press SUBMIT.

Main Screen Layout

The main screen has these features:



Finding An Existing Incident

Vanessa provides a number of convenient ways to locate existing cases:

① **Vanessa IDs.** You can use a four-digit Vanessa ID number. Note the following:

- Historical **CIREN** cases have IDs between 1001-1435.
- Historical **ICAM** cases range from 1436 to about 1806.
- Some historical **VIPA** cases have IDs in the 3000s.
- Historical **VIPA** cases for which data already were present in the old ICAM database in contrast have IDs in the ICAM case range from the 1400s to 1800s.

② **ICAM shared Folder ID.** You can also locate incidents by using an **ICAM** shared folder ID number in the *ICAM-yyyy-[P|B]*nn* format. The search box allows you to find vehicle, pedestrian, and bicyclist cases in this manner.

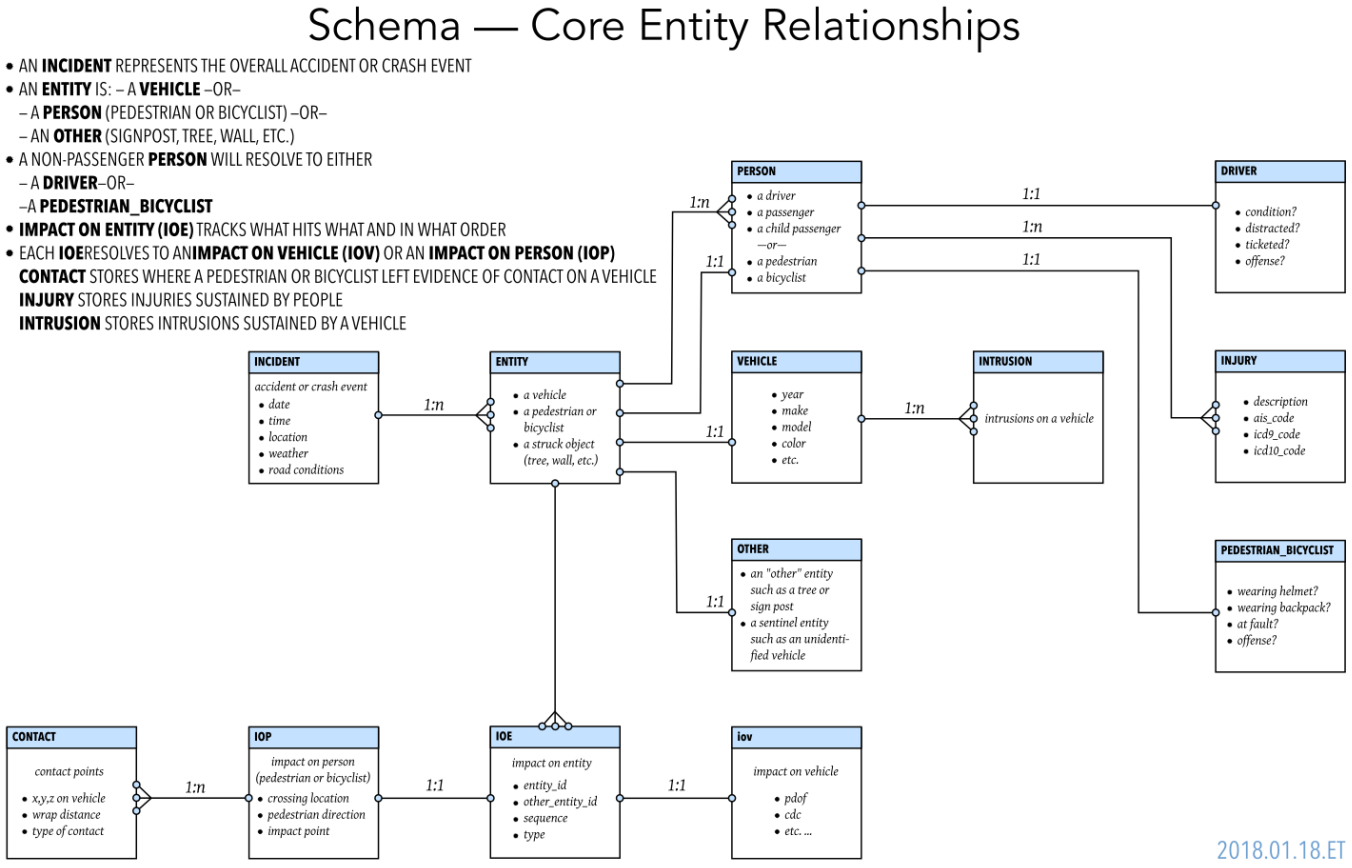
③ **CIREN case ID.** Similarly, you can find historical CIREN cases using the *CIREN-yyyy-nn* pattern.

④ **Legacy ICAM database IDs**. You can also search for a legacy ICAM database ID. To do so, type an "L" followed by a 4-digit ID number, e.g. *Lnnnn*.

⑤ **By Date in ISO format**. Finally, you can also locate cases by date in ISO *yyyy-mm-dd* format. One advantage of using the ISO format when searching is that you can enter partial dates consisting, for example, of just a year and a mont, e.g. *2015-04*, etc.

Logical Layout of the Application

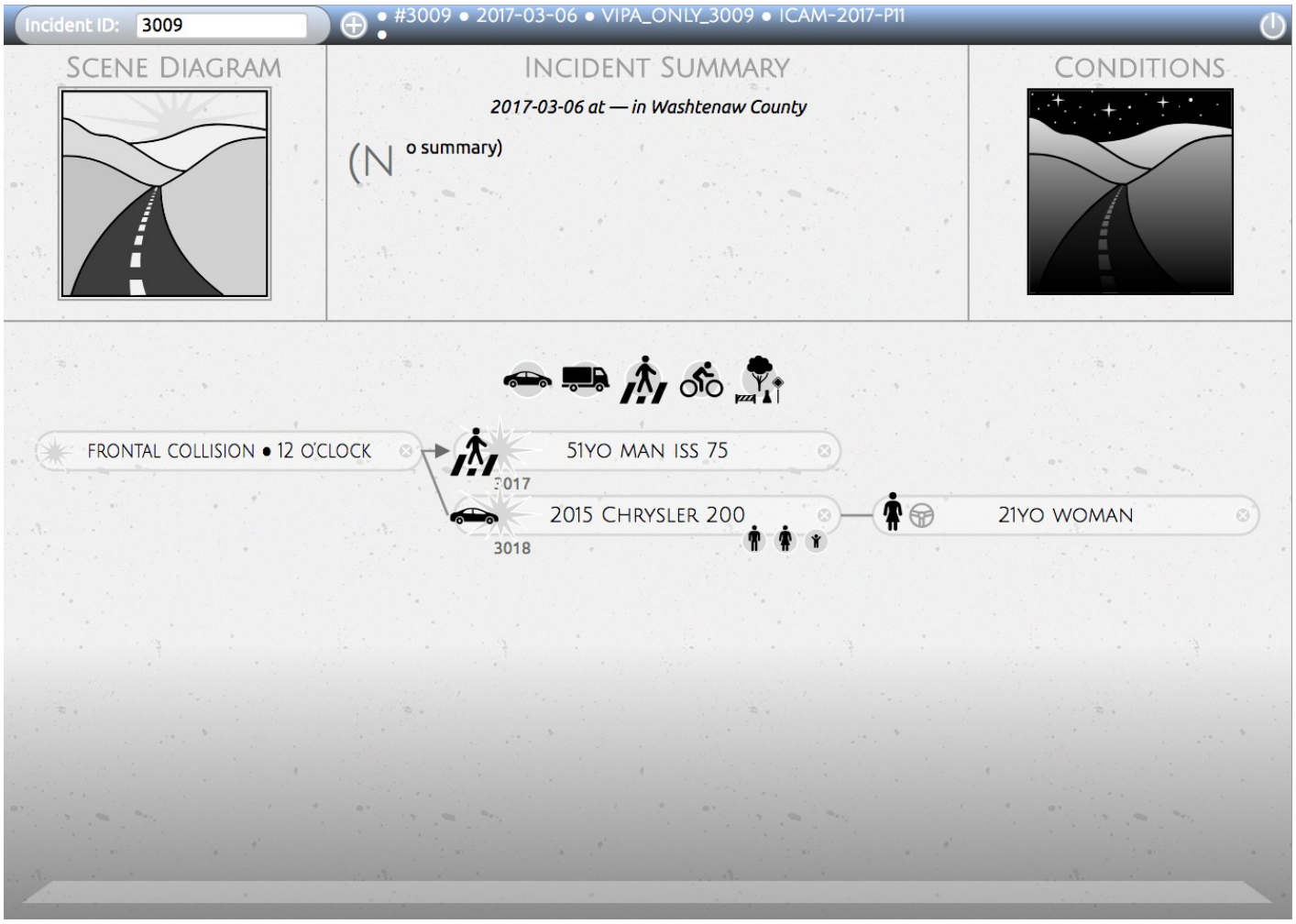
Logical divisions within the application closely follow the Vanessa database schema, which is reproduced below for reference:



In this schema, we conceptually have base classes and derived classes. For example, the *entity* table is considered a base class while *person*s, *vehicle*s and *other* impacted entities can be seen as specific realizations of this abstract *entity* class. The base class *entity* data are not edited directly by the user, but are used extensively by the application. For example, the incident diagram that appears in the lower half of the primary screen (*see example on following page*) is constructed by querying the *entity* table with joins on the specific child tables (*person, vehicle, other*, and so on). When one clicks on a labeled person in the incident diagram, a form appears to edit that person. That form, in addition to having a "save" button for the person data, also has buttons to open up the "injury" form and another button to open up either the "driver" form or the "pedestrian/bicyclist" form, as appropriate. Drivers and pedestrians are specific types of people and specific additional information is recorded for these people (for VIPA cases, *ad minimum*). Thus we see how *driver*s and *pedestrian*s are also conceptual derivitives of the *person* class.

Appearance of a typical pedestrian case

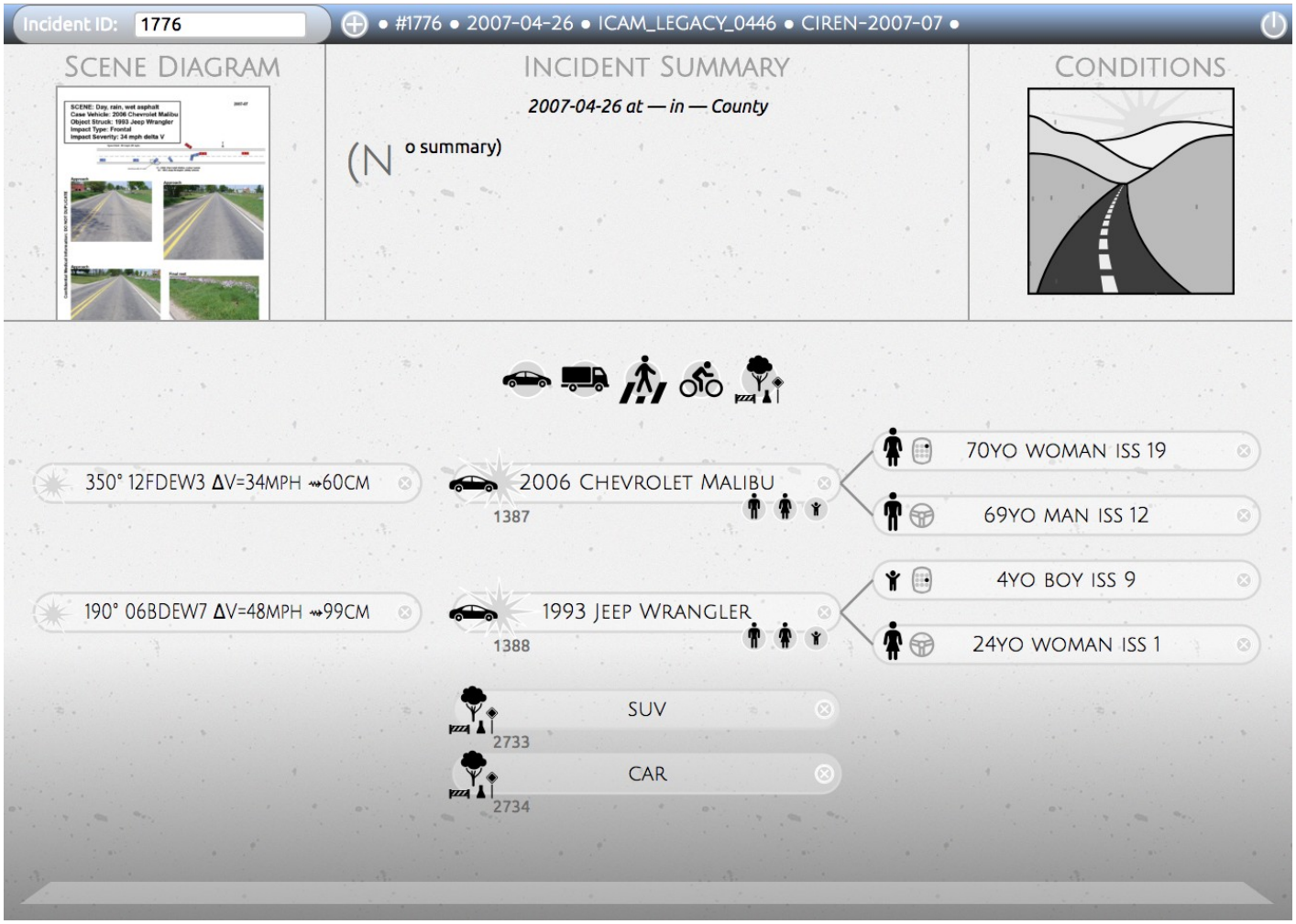
A typical pedestrian case will look something like the example below:



From the visual summary, we quickly grasp that a 21-year-old driver of a 2015 Chrysler hit a 51-year-old pedestrian in a frontal collision, resulting in his death (*ISS=75*).

Appearance of a typical vehicle-vehicle case

Here, for comparison, is what a typical vehicle-vehicle collision might look like (*as of this writing*):



In this example, this example, we see that a Chevrolet Malibu and a Jeep Wrangler hit each other. Drivers and passengers are shown. Also, the associated impacts are shown. However, because this is a legacy case imported from the old database, no collision arrows have been drawn yet. Additionally, we see that an "other SUV" and "other car" are hanging out at the bottom of the diagram. In the legacy ICAM database, we were only able to see that the Malibu hit some other SUV, but we were not able to know specifically *which* SUV. Likewise, in the legacy system, we know that the Jeep Wrangler hit some other car, but without knowing specifically *which* car. These were limitations of the old system. In the new **Vanessa** system, these two separate ICAM cases have been re-united into a single Vanessa incident.

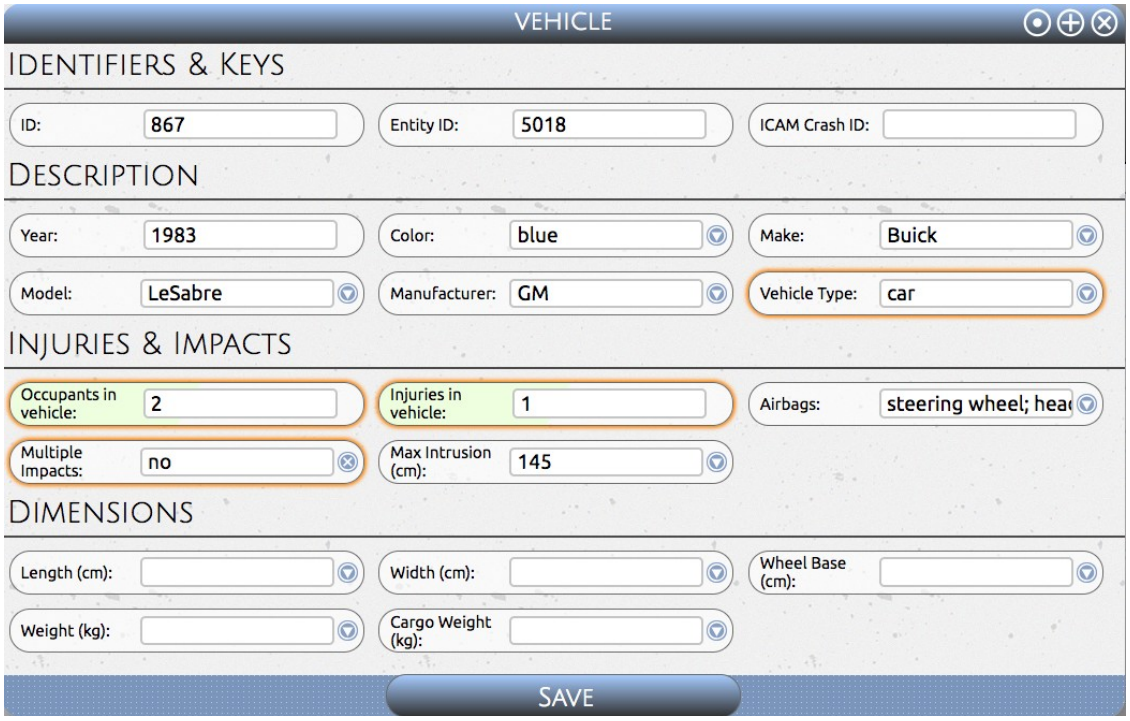
At this point, a human reviewer still needs to edit this incident to: ① remove the now no-longer needed "other SUV" and "other car" from the bottom of the panel, and ② edit the impacts on the left to correctly mark the impacting vehicles. After this has been done, the impact arrows will be drawn correctly.

Making Edits

Clicking on any of the labels in the incident diagram in the lower half of the screen will bring up a form specific to that entity or impact.

The Vehicle Form as an Example

Let's take a look at the *vehicle* form as an example:



At the top of the form is a section entitled *Identifiers & Keys.* Identifiers in this section are often fixed by the application in a read-only form that cannot be edited by the user. In the future, these read-only identifiers may be removed from the display. However, at the current moment when the application is still undergoing a lot of development, displaying these identifiers is quite useful for debugging and for tracing out the historical origin of incidents and cases.

Fields shown above in orange have been changed but not yet saved. In addition to nicely highlighting exactly what has changed, this also provides a visual reminder to the user to click the *save* button before moving on.

Entry of data into the user-interface widgets in general works as described in the V3 Toolkit document, https://icamlinux.surg.med.umich.edu/vanessa/documentation/V3_Documentation.pdf. Since that document was written, the functionality of some user interface widgets have been revised. At the same time, some new user interface widgets have also been written. These improvements are described below.

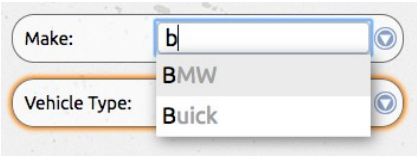
Expanded suggest widgets

Suggest lists are generally appropriate when we want to enter categorical data in a uniform or normalized manner. Suggest lists may differ depending on the number of categories required for any given field. The Vanessa V3 toolkit now has 3 variant forms of suggest list widgets:

① **The original suggest list** which does not have a button to the right of the input box.

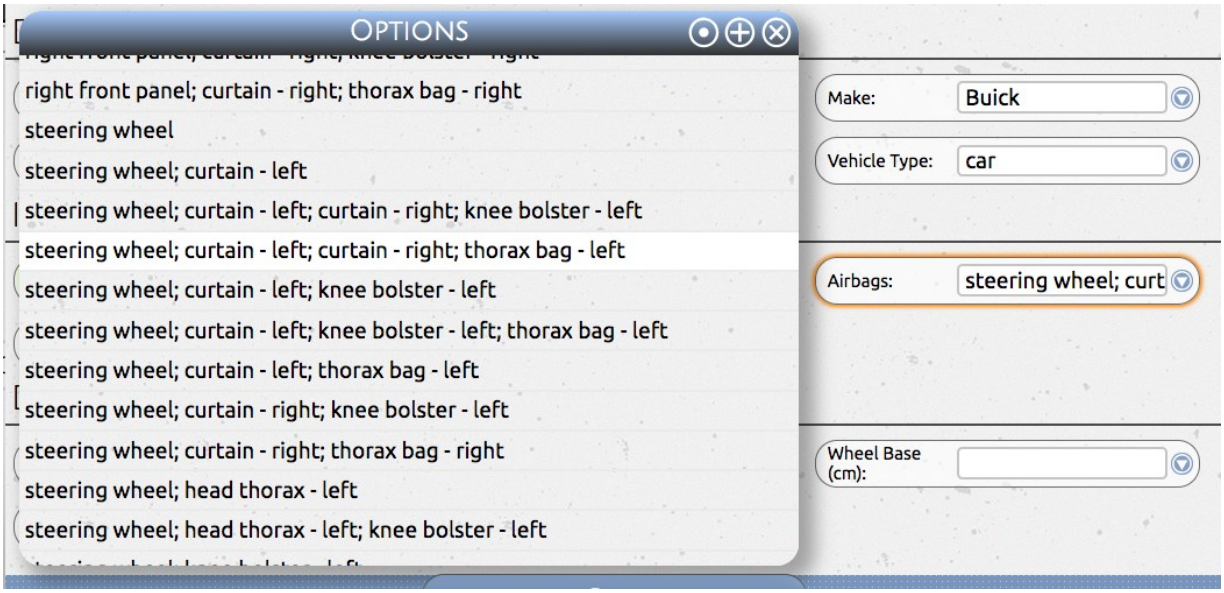
This suggest list is a good choice for categorical data when there are many categories (possibly even hundreds) but where the user is generally very familiar with the options that might be entered. In this case, having a suggest list means the user need not type entries in their entirety. Also, the suggest list helps promote uniform spelling since users are much more likely to now choose from the suggest list rather than type entries in their entirety.

If we start typing the letter "b", a typical suggest widget will look like this:



② **The suggest list with a drop-down button.** This version of the suggest list is most appropriate when there are a small number of possible entries. For example, in the *person* form, there are *—for our data collection and analysis purposes—* only a few different categories of people:

③ **The suggest list with a drop-down button for big lists.** In some cases, there may be a large number of categories present for a field and the user may not be very familiar with all of the options that are possible. In this case, when we press the button for the dropdown list to appear, it might not be possible to display all of the entries at once on screen. In this case, it might be better to present a complete list of possibilities in a separate scrollable list. This is exactly what we do in this new, third variant. Here is an example:



Note on how to implement the big list suggest list

By default, a V3 suggest list will only pull at most *12* unique entries from the database. To implement the "big list" variation, simply override this default by entering *a number greater than the maximum number of unique entries for that category* in the vanessa.sys_lu_data_dictionary table. For example, for the example shown above, we see that there are at present 31 unique entries:

```
sahmdb=# select distinct airbags_legacy from person order by 1;

                                    airbags_legacy
-------------------------------------------------------------------------------------
 Curtain - Left
 Curtain - Left; Curtain - Right
 Curtain - Left; Head Thorax - Left
 Curtain - Left; Thorax Bag - Left
 Curtain - Right
 Head Thorax - Left
 None Deployed/Avail
 Right Front Panel
    .
    .
    .
 Thorax Bag - Left
 Thorax Bag - Right
 Unknown
(31 rows)
```

Therefore, in *sys_lu_data_dictionary*, we enter 50 for max_items:

```
sahmdb=# select id,section_table,table_column,ui_type,match,sort,warn_level,max_items from
sys_lu_data_dictionary where table_column='airbags_legacy';
 id  | section_table |  table_column  |     ui_type      | match | sort | warn_level | max_items
-----+---------------+----------------+------------------+-------+------+------------+-----------
 130 | person        | airbags_legacy | suggest_dropdown |       |      |            |        50
(1 row)
```

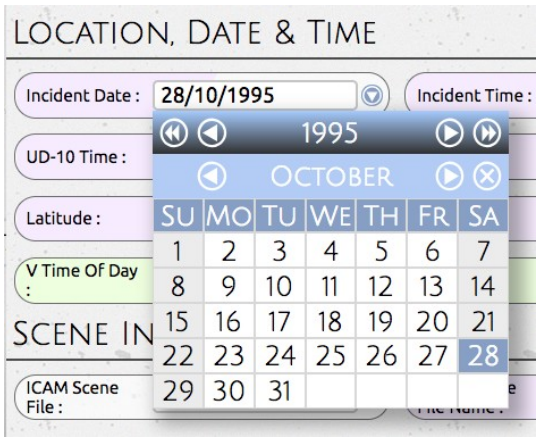
That is all that is required. The V3 toolkit automatically detects when the list returned by the database query is too big to fit on screen in its entirety, and then automatically transfers the list items into a scrollable list inside a popup window.

New Date Entry Widget

Originally the V3 Toolkit was designed to take advantage of the new **HTML5** input types, such as `<input type="date">` (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/date).

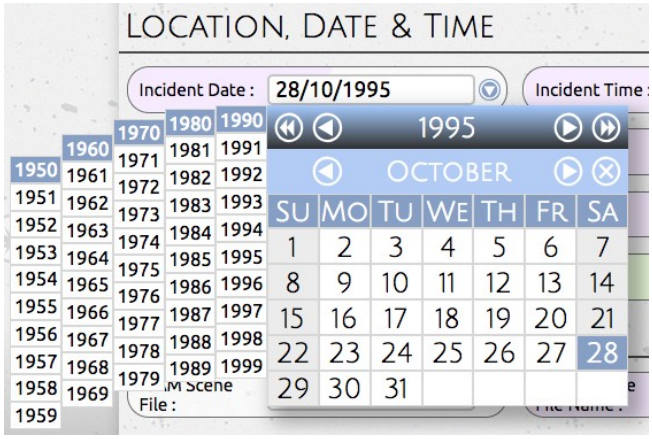
However, the problem is that currently browser support for such **HTML5** input types is uneven and some modern browsers —*such as Safari on Mac OSX*— fallback to a generic `<input type="text">` which is really insufficient for reliable date entry.

I have therefore now added a custom drop-down date widget for the V3 toolkit. The custom widget supports numerous locales and provides several convenience features:



Note that although the widget displays the date in a localized format based on the current locale (*Spanish Latin America ES-419 locale dd/mm/yyyy format shown here*), dates are always transmitted and saved to the database in ISO *yyyy-mm-dd* format, regardless of the local locale.

The date widget also features decade "pop-out" (<<) and (>>) buttons to permit quick entry of things like dates of birth:



New Conversion Widgets

A recurring issue is that the United States continues to use Imperial units in many contexts even though SI (metric) units are used in scientific and engineering contexts, and of course most of the rest of the world also uses SI units in almost all contexts.

ICAM clearly also prefers to use SI units. However, when I looked at legacy ICAM data where both Imperial and SI data were stored, I found numerous discrepancies in the conversions.

To avoid issues with unit conversions going forward, I suggest we remove existing legacy fields demarcated in *miles per hour* (*mph*) or any other non-SI units.

To facilitate data entry, I have added a v3.input.UnitConversion class which makes use of a v3.Converter class to provide a drop-down UI widget for performing bi-directional unit conversions. However note that only SI results get stored in the database.

Currently, I have implemented three converters:

① centimeters <=> inches converter (the default)
② kilograms <=> pounds converter
③ kph <=> mph converter

Additional converters can be added easily by passing custom "*imperialToMetric*" and "*metricToImperial*" methods (*along with appropriate labels*) to the class constructor.

Currently the converters automatically round to whole numbers since most of the relevant columns are specified as integer columns in the database. Obviously, this can be changed or customized as necessary.

Example: a Buick LeSabre has a wheel base of 112.2" . In the dropdown widget, we enter 112 inches, equivalent to 284 centimeters, and we save the requested metric value: