StarCellBio

1. Architecture
2. Code organization
3. Django Server
4. StarCellBio Student's view
5. StarCellBio Course Instructor's view
6. StarCellBio Labnotebook

Architecture

StarCellBio is split in following components:

- Authentication, authorization and data storage on server
- Student's is SPA (Single Page Application) that resides in the browser and communicates with the server only to serialize or de-serialize user session.
- Assignment Builder is SPA that resides in the browser where all interaction happen, including running Studen's view in preview mode.

Authentication, authorization and data storage are implemented as Django application with minimal API exposed to the SPA's.

Student SPA loads including all the code it is necessary to perform all experiments. Code is that driven using JSON data structure. JSON is fed to the business logic encapsulated within /model folder. Business logic exposes underlying data while allowing for intercepting (analytics) or overriding behaviors (allows for relational querying without leaking implementation details to user interface). User interface is implemented in /ui folder as mix of .soy (google closure templates) and .js (javascript) objects.

Student SPA uses simple MVC, event driven architecture. All events are registered upon initialization. Event handlers interact with model and request page to be re-rendered by changing URL hash or by invoking .repaint() method.

User interface is build in such fashion that each HTML element carries sufficient information to determine which handler to call and which part of the model it impacts. Event handler is determined by class associated with the HTML element. "scb_f_" classes represent handlers to be invoked, while "scb_s_" classes represent element styling. This allows for very clean and DRY code that allows event handlers to be reused at different locations in the application as needed.

StarCellBio uses RESTful URL's to represents its state in the browser, and browser URL fully defines which view will be rendered. URL contains parameter view. As a matter of coding standard this name is the same as name of the file that renders this

view in /ui folder. And name of template is the same name as view with .soy extension. All the view's methods are in the scb namespace with the view name, and scb name space contains class that implements at least one methods 'show' that is invoked to render this view. Namespace also contains static method register with registers all required handlers for the view (scb_f_ classes). This structure allows for easy code navigation and any new code should follow these convetions.

Data structure on the client (browser)

Top level object is assignment list represented by AssignmentList, this is defined in Assignments.js. AssignmentList is defined by common list code (described later). Each element of the list is represented by Assignment object as described in Assignments.js.

Assignment

Assignment represent complete assignment, including owning all of experiments in assignment, notebook and it has metadata that allows it to render in UI.
Assignment uses Common Entry Code template, and adds following fields:

- Notebook (scb.Notebook)
- Experiments (scb.ExperimentList)
- Template (JSON map representing assignment)
- Course (string - key)
- Course_Name (string – descriptive name)
- last_instruction
- operation
- permission
- template_id
- students
- sort
- is_new_assignment
- is_new_course
- has_background_bands
- background_band_list
- course_prepared
- assignment_prepared
- has_temperature
- has_start_time
- has_duration
- has_collection_time

ExperimentList

ExperimentList uses Common List template, and adds following two methods:

- Start – return new experiment and register it with Assignment

- Next_name(to_increment) – returns name of next experiment

Experiment

Experiment represents single experiment in an assignment. Assignment can have many experiments. In UI it is represented by second tab, and individual one can be selected from the pulldown.
Experiment uses Common Entry Code and adds following fields:

- hypothesis (string)
- objective (string)
- technique (string)
- cell_treatment_list (scb.CellTreatmentList)
- western_blot_list (scb.WesternBlotList)
- facs_list (scb.FacsList)
- microscopy_list (scb.MicroscopyList)
- last_view (defaults to experiment_design)
- prev_step
- last_step
- last_scroll
- last_technique
- last_id
- last_technique_view
- last_param
- setup_finished (bool)
- setup_visible (bool)
- new_row (current state of new row)
- design_wb_cb
- design_fc_cb
- design_mi_cb
-

CellTreatmentList

CellTreatmentList is part of an experiment, uses Common List template, and adds following two methods:

- start
- duplicate

CellTreatment

CellTreatment is instance of cell treatment, and uses Common Entry Code and adds following fields:

- cell line
- treatment_list (scb.TreatmentList)
- stimulation time
- collection schedule list (scb.CollectionScheduleList)

TreatmentList

TreatmentList is part of CellTreatment, it uses Common List template and wrapper for Treatment. It also adds following fields:

- first
- start

Treatment

Treatment and uses Common Entry Code and adds following fields:

- drug_list (scb.DrugList)
- temperature (string)
- collection_id
- microscope
- facs
- schedule_value
- duration_value
- schedule
- duration

DrugList

DrugList is part of Treatment, it uses Common List template. It also adds following methods:

- start_default
- start
- set_list

Drug

Drug uses Comon Entry Code and adds following fields:

- drug_id
- concentration_id
- collection_id
- drug_name (derives data from template and drug_id)
- drug_concentration (derives data from template and concentration_id)

CollectionScheduleList

CollectionSheduleList uses CommonList code without extensions.

CollectionSchedule

CollectionSchedule uses Common Code template and adds only two fields:

- schedule_value (in seconds)
- schedule (string)

WesternBlotList

WesternBlotList uses Common List Code and adds following functionality:

- start_tabs_index (used by WB UI)
- start

- new_using_making_lysates
- duplicate

WesternBlot

WesternBlot uses Common Code template and adds:

- lysate_prepared
- marker_loaded
- wells_loaded
- gel_type
- is_transfered
- lanes_list (scb.WesternBlotLaneList)
- gel_list (scb. WesternBlotGelList)
- last_gel
- canvas_metadata
- prep_scroll
- is_cell_treatment_enabled
- samples_show_state
- measure_show_state
- rows_state
- rows_state_count

WesternBlotExposureList

WesternBlotExpoureList uses Common List code.

WesternBlotExposure

WesternBlotExposure uses Common Code template and adds:

- canvas_data
- schedule_value
- schedule

WesternBlotGelList

WesternBlotGelList uses Common List code.

WesternBlotGel

WesternBlotGel uses Common Code template and adds:

- primary_anti_body
- secondary_anti_body
- exposure_time
- is_developed
- canvas_data
- canvas_metadata
- marks

WesternBlotLaneList

WesternBlotLaneList uses Common List code, and adds:

- start
- duplicate
- reorder
- cell_treatment_id (as grouped list)

WesternBlotLane

WesternBlotLane uses Common Code template and adds:

- kind
- ip
- marks
- cell_treatment_id
- collection_schedule_id
- order_id
- experiment_id
- making_lysate_id
- experiment
- cell_treatment
- collection_schedule
- kinds

FacsList

FacsList uses Common List code, and adds:

- start_tabs_index
- start
- new_using_making_lysates
- duplicate

Facs

Facs uses Common Code template and adds:

- lanes_list (scb.FacsLaneList)
- samples_finished
- sample_prepared
- sample_analysis
- double_analysis
- gate_count
- midpoint
- prep_scroll
- show_analysis
- apply_dna_analysis_to_all
- instructions_show_state
- samples_show_state
- lane_selected
- selected_lane
- is_cell_treatment_enabled
- rows_state

FacsLaneList

FacsLaneList defined:

- list
- ensure_experiment_lanes_for_experiment
- duplicate
- reorder
- filter
- reset
- length
- cell_treatment_id

FacsLane

FacsLine uses Common Code template and adds:

- kind
- conditions
- cell_treatment_id
- gates_id
- exp_id
- bisector_gate_created
- canvas_metadata
- canvas_metadata_analysis
- gate_selected
- experiment_id
- selected_gate
- experiment
- cell_treatment
- collection_schedule
- kinds
- display_text

FacsLanePreparationList

FacsLanePreparationList uses Common List template and adds:

- start
- duplicate
- reorder
- filter
- reset

FacsLanePreparation

FacsLanePreparation uses Common Code template and adds:

- kind
- treatment
- sub_treatment

- chart_data

MakingLysateList

MakingLysateList uses Common List template and adds:

- start
- new_using_experiment

MakingLysate

MakingLysate uses Common Code template and adds:

- finished
- lysate_kind_list

MicroscopyList

MicroscopyList uses Common List template and adds:

- start_tabs_index
- start
- new_using_making_lysates
- duplicate

Microscopy

Microscopy uses Common Code template and adds:

- slide_prepared
- lanes_list
- samples_finished
- lane_selected
- is_cell_treatment_enabled
- warning_fired
- laser_on
- red_enabled
- blue_enabled
- green_enabled
- merge_enabled
- scroll
- prep_scroll
- samples_show_state
- navigation_show_state
- enable_samples
- disable_blur
- disable_brightness
- light_on
- selected_lane
- rows_state

MicroscopyLaneList

MicroscopyLaneList defines:

- list
- start
- ensure_experiment_lanes_for_experiment
- duplicate
- remove
- get
- reorder
- filter
- reset
- length
- cell_treatment_id
-

MicroscopyLane

MicroscopyLane uses Common Code template and adds:

- kind
- lens_map
- slide_conditions
- cell_treatment_id
- current_slides
- mag
- experiment_id
- experiment
- cell_treatment
- collection_schedule
- kinds
- display_text

Notebook

Notebook uses Common Code template and defines:

- scroll
- edit_text
- edit_image
- sections (scb.NotebookSectionList)
- image_experiment_id
- image_western_blot_id
- image_western_blot_gel_id
- image_facs_id
- image_facs_lane_id
- image_microscopy_id
- image_microscopy_lane_id
- section_selected
- selected_section
- selected_experiment

- selected_western_blot
- selected_western_blot_gel
- selected_facs
- selected_facs_lane
- selected_microscopy
- selected_microscopy_lane

NotebookElementList

NotebookElementList uses Common List code.

NotebookElement

NotebookElement uses Common Code template and defines:

- type
- data
- view
- experiment_id
- headings
- rows
- western_blot_id
- gel_id
- exposure_time
- facs_id
- facs_lane_id
- microscopy_id
- microscopy_lane_id
- selected_experiment
- selected_western_blot
- selected_western_blot_gel
- selected_facs
- selected_facs_lane
- selected_microscopy
- selected_microscopy_lane

NotebookSectionList

NotebookSectionList uses Common List template.

NotebookSection

NotebookSection uses Common Code template and adds:

- hypothesis
- objective
- technique
- order_id
- elements (scb.NotebookElementList)

Common List

Common list code is implemented as common template for business logic. It is defined in ModelHelpers.js and is used by all *List objects in model/ folder. It defines following fields:

- List – as an array
- Selected_id – last selected element of the list
- Selected – object instantiated from the prototype representing element of the list
- Get(id) – get element by ID (search list for it)
- Remove(id) – remove element from the list by ID
- Length – size of the list

Common Entry Code

Common Entry Code represents what each object in the model is guaranteed to have. It is defined in ModelHelper.js and is used by all non-list objects in the model. It defines following fields:

- Id – unique ID
- Name – display name
- Description – description as needed
- Created_at – timestamp when object was first created

Template

Template is free-form JSON template that is utilized to drive StarCellBio application and to contain all information that course instructors customize per assignment. Following fields are part of template:

- instructions
- ui
- experiment_setup
- cell_lines
- drugs
- lysate_kinds
- primary_anti_body
- secondary_anti_body
- concentrations
- micro_kinds
- slides
- slide_parser
- model

Template (InstructionsList)

InstructionList is an array of objects that contain title and text

Template (UI)

Is parent element has sub-documents:

- experimental_design
- experiment_setup
- microscopy
- add_multiple_dialog
- western_blot

Template (ui experiment_design)

Has following elements:

- array of techniques
- array of gel_types

Template (ui experiment_design)

Has following elements:

- table
- actions

Template (ui microscopy)

Has following elements:

- disable_brightness
- disable_blur

Template (ui westernblot)

Has following elements:

- format
- keys

Template (cell_lines)

CellLines is dictionary consisting of CellLine objects referenced by id.
CellLine object has name.

Template (drugs)

Drugs is dictionary consisting of Drug objects referenced by id.
Drug object has name and array of applicable concentrations.

Template (LysateKinds)

LysateKinds is dictionary consisting of LysateKind objects referenced by id.

Template (PrimaryAntiBodies)

PrimaryAntiBodies is dictionary consisting of PrimaryAntiBody objects referenced by id.
PrimaryAntiBody object has:

- name
- secondary
- marks
- gel_name

Template (SecondaryAntiBodies)

SecondaryAntiBodies is dictionary consisting of SecondaryAntiBody objects referenced by id.
SecondaryAntiBody object has name

Template (Concentrations)

Concentrations is dictionary consisting of Concentration objects referenced by id.
Concentration object has name and value.

Template (MicroscopyKinds)

MicroscopyKinds is a dictionary consisting of MicroscopyKind objects referenced by id.
MicroscopyKind object has name and conditions (MicroscopyKindConditions)

Template (MicroscopyKindConditions)

MicroscopyKindConditions is a dictionary consisting of MicroscopyKindConditionobjects referenced by id.
MicroscopyKindCondition object has name and short_name

Template (Slides)

Slides is a dictionary that is used by Microscopy model to determine which image to display in microscopy UI.

Template (Model)

Model contains models that apply rules to the data provided. There are models for all techniques. Usually there is a default model and possibly few specialized models. These models are yet to be documented.

Data structure on the server (django/mysql)

Structures to support StarCellBio runtime and that will be used by Assignment builder are:

- Course
- Assigment
- Student Assignment
- UserCourse

Assignment

Assignment is a central concept in StarCellBio. This is expressed as a database model with the following characteristics:

- courseID – foreign key linking it to Course
- assignmentID – primary key
- assignmentName – name of the assignment (same as value in JSON object)
- data – this is JSON dump of the assignment object, used by assignment builder and by runtime
- ownerID – owner of this course
- access – is this course public, private, archived,…

Course

Course is naming concept in StarCellBio, it is used to group courses for easy navigation. It's model only has `code` used to register for course and `course_name` used to display course in UI (same as value in JSON object)

StudentAssignment

StudentAssignment is realization of the assignment with embedded student data in it. The model has:

- student – owner of the student assignment
- course – course this assignment is associated with
- assignmentID, assignmentName – assignmentID to which this assignment relates
- token – token of currently active client (browser) to ensure that there are no data collisions in running data from multiple browsers on the same time
- data – JSON dump of the assignment object from student's browser

UserCourse

UserCourse is mapping between logged in user and the course they are enrolled in.

Other tables

On top of above mentioned tables StarCellBio uses standard Django user management. It also allows logging in with other oauth providers using allauth module.

Server side URLS

StarCellBio exposes set of views to the user these URLs are:

- index.html main page to load all the SCB runtime
- scb/contact – views.contact – sends email to starcellbio@mit.edu email
- scb/get_model.js – views.get_model – returns model from the server that is appropriate for the user (authenticated and guest)
- scb/ get_student_courses.js - views.get_student_courses - This view gets the courses for a student for their account. For the instructor, it gets the courses it can view
- scb/get_instructor_assignments.js - views.get_instructor_assignments – get list of assignments instructor can view
- scb/edit_assignment.js – views.edit_assignment
- scb/create_course.js – views.create_course
- scb/create_new_assignment.js – views.create_new_assignment
- scb/get_user.js – views.get_user
- scb/post_state.js – views.post_state – save student state

Assignment builder uses tastypie (https://django-tastypie.readthedocs.org/en/latest/) to create rest interface for resources it needs to access. Resources are defined in backend/services.py.
At this time following resources are defined:

- UserResource
- CourseResource
- AssignmentResource
- StudentAssignmentResource

These resources expose REST API towards above mentioned data models.

Course Instructor View

As Course Instructor View is not yet written, list of use cases will be provided here.

Use case 1: Instructor login

Upon logging in instructor is presented with a dashboard screen listing instructor's assignments. From dashboard instructor can choose to:

1. Edit existing assignment
2. Edit courses

3. Preview existing assignment
4. Create new assignment

Use case 2: Edit Courses

When instructor chooses to edit courses, instructor can:

1. Edit existing course
2. Add new course
3. Delete course if no assignments that are using it

Use case 3: Add new course or edit course

Course consists of:

- name
- course code

Course edit updates live version. Course code should not change after it is assigned during creation and if students are using the course.

Updates to “course” and “course_name” fields

Use case 4: Preview existing assignment

Previewing existing assignment should display assignment, as it would appear to student. This is done by creating *scb.MainFrame* with *master_model* containing only assignment that needs to be previewed.

Use case 5: Create new assignment

Creating assignment is same as editing assignment. The user needs to provide minimal metadata before assignment can be saved.
Minimal metadata is:

- course
- assignment name
- assignment code

Updates: “course”, “name”, “id” fields

Use case 6: Edit assignment

Editing assignment is complex process as data stored in backend.models.Assignment.data should be kept in sync with the user interface.

Assignment editing allows user to select to edit:

- Experiment setup
- Western blot
- Flow cytometery
- Microscopy

Updates:

"template.ui.techniques"

Use case 7: Edit experiment setup

Setting up experiment setup consists of selecting to edit:

- Experiment setup text
- Strains
- Protocols
- Which strain/protocol combinations are enabled

This updates various elements of the template in particular:

"template.instructions",
"template.strains",
"template.ui.experiment_setup.table",
"template.ui.experiment_setup.add_multiple_dialog",
"template.concentrations",
"template.drugs",
"template.experiment_temperatures",
"template.cell_lines",
"template.time_unit"

Use case 8: Edit experiment setup - strains

Strains editing is CRUD process.

Strains contain key and name that are stored in "template.strains" and used by various other portions of template.

Use case 9: Edit experiment setup - protocols

Each protocol is an array of:

- treatment
- concentration
- concentration unit
- temperature
- start time
- duration

This updates various elements of the template in particular:

"template.ui.experiment_setup.add_multiple_dialog",
"template.concentrations",
"template.drugs",
"template.experiment_temperatures",
"template.cell_lines",
"template.time_unit"

Use case 10: Edit experiment setup – strain protocol map

Strain protocol map describes which combination of strains work with which protocol. Thus it is many-to-many relationship between protocols and strains. This is done so that it is easier to instructor to create larger assignments without need to repeat information

This updates:

"template.ui.experiment_setup.add_multiple_dialog",

Use case 11: Edit western blot

Western blot editor consists of setting western blot metadata which include:

- lysate kinds
- gel types

Also, western blot has list of primary and secondary anti-body combination.

This updates:

"template.lytate_kinds"
"template.ui.experimental_design.gel_types"

Use case 12: Edit western blot – antibody / strain-protocol

For each strain-protocol that was enabled in experiment setup user is presented with antibody combinations and depending on lysate kinds enabled should specify weights and intensities, as they will appear on western blot.

This updates:

"template.model.western_blot"

Use case 13: Edit Flow Cytometery

Flow cytometery editor starts by allowing instructor to specify treatment, analysis and condition that apply for sample preparation.

Use case 14: Edit Flow Cytometery – preparation / strain-protocol

For each preparation / strain-protocol combination instructor will be able to specify graph that will appear. This can be from the list of templates that can be customized or by drawing histogram.

This updates:

"template.model.facs"

Use case 15: Edit histogram

Histogram tool should allow instructor to add point and smooth curve. This can be used to create new templates or to adjust how histogram appears to the student.

This will need to be developed as new model of model FACS to support arbitrary date, right now 7 types are supported:

Normal
s-block'),
g1-block'),
g2-block'),
alpha-block'),
2-peak-normal-400
peak-100-normal-400
2-peak-uneven-normal-400
peak-50-normal-400
4-peak-normal-400
s-block-normal-400

Use case 16: Edit Microscopy

In preparing microscopy instructor creates an array, a table, of sample preparation techniques. Instructor selects if sample preparation is from fixed or live cells, which analysis to use, dye/stain or anti-body labeling, and finally which conditions to apply to the sample.

Use case 16: Edit Microscopy – preparation / strain-protocol

After that instructor selects for each combination of treatment preparation and enabled strain-protocol combination a slides to be displayed and parameters used to configure microscopy treatment.

Example assignment template

This is an example of template that assignment builder will produce. Below are described elements and how they match to use cases above.

```
{
  id: 'usability_test',  // assignmentID
  name: 'SCB Usability Test', // assignmentName
  course: 'StarX', // courseID
  course_name: 'Prototypes', // course name
  description: "Placeholder", // course description
  notebook: {},  // other initial data elements, normally empty
  experiments: {}, // other data elements, normally empty
  template: { // template dictionary
    instructions: [ // instructions array, each element is a tab
      ['Placeholder', 'Usability test'] // 1st subelement is title, 2nd is content
    ],
    ui: {
```

```
        experimental_design: {
          techniques: [ 'wb' , 'facs' ] // enabled techniques
        },
        experiment_setup: {table: [ // describes experiment setup table
          {kind: "cell_plate",
            title: " ",
            editable: false},
          {kind: "cell_line",
            title: "Strain",
            editable: false
          },
          {kind: "treatments",
            children: [
              {kind: "drug", title: "Treatment", editable: true},
              {kind: "concentration", title: "Concentration", editable: true},
              {kind: "start", title: "Start", editable: false},
              {kind: "duration", title: "Duration", editable: false}
            ]
          },
          {kind: "actions",
            title: "Actions"
          }
        ], actions: [
        ]
        },
        western_blot: {format: "%CELL_LINE%, %TREATMENT%,
%CONCENTRATION%", // describes formatting in western blot, reused in facs
          keys: {
            '%CELL_LINE%': {attr: ['cell_line'], map: ['cell_lines', '%KEY%', 'name']},
            '%TREATMENT%': {attr: ['treatment_list', 'list', '0', 'drug_list', 'list', '0',
'drug_id'], map: ['drugs', '%KEY%', 'name']},
            '%CONCENTRATION%': {attr: ['treatment_list', 'list', '0', 'drug_list', 'list',
'0', 'concentration_id'], map: ['concentrations', '%KEY%', 'name']}
          }
        }
      },

      experiment_setup_actions: { // describes list of protocols
        cell_lines: [
          {
            id: 'wt',
            title: 'Wild Type',
            cell_line: 'wt'
          }
        ],
        treatment_protocol_list: [
```

```
{
   id: 'P1',
   title: 'Buffer Only',
   treatment_list: {list: [
      {schedule_value: 0, schedule: 'immediately', // start
         duration_value: 3600 * 24 * 3, duration: '3 d', // end
         drug_list: {list: [
            {drug_id: 'nc', concentration_id: 0}
         ]}}
   ]}
},
{
   id: 'P2',
   title: 'V1 low conc',
   treatment_list: {list: [
      {schedule_value: 0, schedule: 'immediately', // start
         duration_value: 3600 * 24 * 3, duration: '3 d', // end
         drug_list: {list: [
            {drug_id: 'nc', concentration_id: '0'},
            {drug_id: '1', concentration_id: '1'}
         ]}}
   ]}
},
{
   id: 'P3',
   title: 'V1 high conc',
   treatment_list: {list: [
      {schedule_value: 0, schedule: 'immediately', // start
         duration_value: 3600 * 24 * 3, duration: '3 d', // end
         drug_list: {list: [
            {drug_id: 'nc', concentration_id: '0'},
            {drug_id: '1', concentration_id: '125'}
         ]}}
   ]}
},
{
   id: 'P4',
   title: 'V2 low conc',
   treatment_list: {list: [
      {schedule_value: 0, schedule: 'immediately', // start
         duration_value: 3600 * 24 * 3, duration: '3 d', // end
         drug_list: {list: [
            {drug_id: 'nc', concentration_id: '0'},
            {drug_id: '1', concentration_id: '50'}
         ]}}
   ]}
```

```
        },
        {
          id: 'P5',
          title: 'Many drugs',
          treatment_list: {list: [
            {schedule_value: 0, schedule: 'immediately', // start
              duration_value: 3600 * 24 * 3, duration: '3 d', // end
              drug_list: {list: [
                {drug_id: 'nc', concentration_id: '0'},
                {drug_id: '1', concentration_id: '0'},
                {drug_id: '2', concentration_id: '5'},
                {drug_id: '3', concentration_id: '10'}
              ]}}
          ]}
        },
        {
          id: 'P6',
          title: 'Many drugs, Many times',
          treatment_list: {
            list: [
              {schedule_value: 0, schedule: 'immediately', // start
                duration_value: 3600 * 24 * 3, duration: '3 d', // end
                drug_list: {list: [
                  {drug_id: 'nc', concentration_id: '0'},
                  {drug_id: '1', concentration_id: '0'},
                  {drug_id: '2', concentration_id: '5'},
                  {drug_id: '3', concentration_id: '10'}
                ]}}
            ]}
        }
      ],
      collection_schedule_list: [
        {id: '3 d', title: '3 days'}
      ]
    },

    add_new_row_instructions: 'On this page, set up your experiment to treat the wild-type worms with the four new drugs, Vulvarines 1-4, identified in your chemical screen. <ul><li>To get started, click <b>Add Treatment Protocol.</b></li><li>For each treatment protocol, select the <i>C. elegans</i> strain, treatment(s), and treatment dose.</li><li> For all of your treatments, treat the <i>C. elegans</i> immediately (time = 0 minutes) and collect after 3 days.</li><li>Once you finish setting up your experiment, select <b>Finish setup & run experiment.</b> After you run your experiment, you will be unable to change your treatment protocols.</li></ul>', // text for add_new_row
```

```
concentrations: { // defined concentrations
  '1': { // ‘1’ is key
    name: '1 ' + microEntity + 'M', // display tag
    value: 1000 // value used by models
  },
  '5': {
    name: '5 ' + microEntity + 'M',
    value: 5000
  },
  '10': {
    name: '10 ' + microEntity + 'M',
    value: 10000
  },
  '20': {
    name: '20 ' + microEntity + 'M',
    value: 20000
  },
  '25': {
    name: '25 ' + microEntity + 'M',
    value: 25000
  },
  '40': {
    name: '40 ' + microEntity + 'M',
    value: 40000
  },
  '80': {
    name: '80 ' + microEntity + 'M',
    value: 80000
  },
  '125': {
    name: '125 ' + microEntity + 'M',
    value: 125000
  },
  '10n': {
    name: '10 nM',
    value: 10
  },
  '50': {
    name: '50 nM',
    value: 50
  },
  '100': {
    name: '100 nM',
    value: 100
  },
  '200': {
```

```
      name: '200 nM',
      value: 200
    },
    '400': {
      name: '400 nM',
      value: 400
    },
    '0': {
      name: '0 nM',
      value: 0
    }
},
drugs: { // list of drugs used in protocols
    'nc': { // ‘nc’ is key
      name: 'Buffer only', // display label
      concentrations: [0] // valid concentrations
    },
    '1': {
      name: 'Vulvarine 1',
      concentrations: [5, 10, 20, 40, 80]
    },
    '2': {
      name: 'Vulvarine 2',
      concentrations: [50, 100, 200, 400]
    },
    '3': {
      name: 'Vulvarine 3',
      concentrations: [1, 5, 25, 125]
    },
    '4': {
      name: 'Vulvarine 4',
      concentrations: ['10n', 50, 100, 200, 400]
    }
},
experiment_temperatures: { // list of temperatures
    '25': { // ‘25’ is key used by protocols
      name: "25" + degreeEntity + "C" // label
    }
},
cell_lines: { // list of cell lines
    'wt': { // cell line key
      name: 'Wild Type' // cell line display label
    }
},
time_unit: { // time unit used
    kind: 'minutes'
```

```
},
primary_anti_body: { // primary anti-body list used by western_blot
   order: [1, 2, 3, 9, 4, 5, 6, 7, 8, 9], // display order in pull-downs
   1: { // primary anti-body key
      name: 'rabbit anti-let-23', // primary anti-body label
      secondary: [1], // secondary anti-body list that matches primary anti-body
      marks: [ // list of default marks for western blot
         {weight: 24, intensity: .11},
         {weight: 36, intensity: .4},
         {weight: 48, intensity: .04}
      ],
      gel_name: 'let-23' // title for gel for this anti-body in western blot
   },
   2: {
      name: 'mouse anti-let-60',
      secondary: [3],
      marks: [
         {weight: 48, intensity: .04}
      ],
      gel_name: 'let-60'
   },
   3: {
      name: 'goat anti-lin15A',
      secondary: [2],
      marks: [
         {weight: 12, intensity: .02}
      ],
      gel_name: 'lin15A'
   },
  4: {
      name: 'goat anti-lin-1',
      secondary: [2],
      gel_name: 'lin-1'
   },
   5: {
      name: 'mouse anti-Dpy-5',
      secondary: [3],
      gel_name: 'Dpy-5'
   },
   6: {
      name: 'rabbit anti-Lon-2',
      secondary: [1],
      gel_name: 'Lon-2'
   },
   7: {
      name: 'mouse anti-Sma-4',
```

```
      secondary: [3],
      gel_name: 'Sma-4'
    },
    8: {
      name: 'goat anti-Unc-22',
      secondary: [2],
      gel_name: 'Unc-22'
    },
    9: {
      name: 'rabbit anti-tubulin',
      secondary: [1],
      gel_name: 'anti-tubulin',
      marks: [
        {weight: 50, intensity: 25.1}
      ]
    }
},
secondary_anti_body: { // list of secondary anti-bodies
    1: { // anti-body key
      name: 'donkey anti-rabbit' //display label
    },
    2: {
      name: 'rabbit anti-goat'
    },
    3: {
      name: 'goat anti-mouse'
    }
},
lysate_kinds: { // list of lysate kinds
    'whole': { // key
      name: 'Whole Cell' // display label
    },
     'cyto':{
     name:'Cytoplasm'
     },
     'nuclear':{
     name:'Nuclear'
     }
},
model: {// model
    western_blot: {// this applies to western blot
      'cyto': {// it acts on cytoplasm (thus on whole cell lysate as well)
        'parser_fixed': [
          {
            'cell_line': 'wt', // matching cell line
            'transfer_function': 'delta', // matching transfer functoin
```

```
    'drug': 1, // Vul 1
    'cutoff': 10000, // delta cut-off
    'above_marks': [ // if drug concentration above & matches
      {
        name: 'let-23',
        weight: 150,
        intensity: -40,
        primary_anti_body: [1]
      }
    ],
    'below_marks': [] // if drug concentration below & matches
  },
  {
    'cell_line': 'wt',
    'transfer_function': 'delta',
    'drug': 2, // Vul 2
    'cutoff': 200,
    'above_marks': [
      {
        name: 'let-60',
        weight: 21,
        intensity: -100,
        primary_anti_body: [2]
      }
    ],
    'below_marks': []
  },
  {
    'cell_line': 'wt',
    'transfer_function': 'delta',
    'drug': 3, // Vul 3
    'cutoff': 25000,
    'above_marks': [
      {
        name: 'let-15A',
        weight: 79,
        intensity: -100,
        primary_anti_body: [3]
      },
      {
        name: 'let-15B',
        weight: 163,
        intensity: -100,
        primary_anti_body: [9]

      }
```

```
    ],
    'below_marks': []
  },
  {
    'cell_line': 'wt',
    'transfer_function': 'delta',
    'drug': 4, // Vul 3
    'cutoff': 400,
    'above_marks': [
      {
        name: 'let-1',
        weight: 48,
        intensity: -100,
        primary_anti_body: [4]
      }
    ],
    'below_marks': []
  },
  {
    'cell_line': 'wt',
    'transfer_function': 'static',
    'marks': [
      {
        name: 'let-23',
        weight: 150,
        intensity: 40,
        primary_anti_body: [1]
      },
      {
        name: 'let-60',
        weight: 21,
        intensity: 100,
        primary_anti_body: [2]
      },
      {
        name: 'let-15A',
        weight: 79,
        intensity: 100,
        primary_anti_body: [3]
      },
      {
        name: 'let-15B',
        weight: 163,
        intensity: 100,
        primary_anti_body: [9]
      },
```

```
                    {
                      name: 'let-1',
                      weight: 48,
                      intensity: 100,
                      primary_anti_body: [4]
                    },
                    {
                      name: 'Dpy-5',
                      weight: 20,
                      intensity: 60,
                      primary_anti_body: [5]
                    },
                    {
                      name: 'Lan-2',
                      weight: 100,
                      intensity: 40,
                      primary_anti_body: [6]
                    },
                    {
                      name: 'Sma-4',
                      weight: 75,
                      intensity: 25,
                      primary_anti_body: [7]
                    },
                    {
                      name: 'Unc-22',
                      weight: 40,
                      intensity: 10,
                      primary_anti_body: [8]
                    }
                  ]
                }
              ]
            }
          }
        }
      }
    };
```

UI Sketches

User interface sketches supporting use cases are available in dropbox and in trello.

- (•) Create new assignment
- ( ) Use an existing assignment as a template