vis
functional spec
I think this needs to be updated did you implement the reward? if not, you should remove it from the spec
don't understand what "poll" means in the options use case

is the authoring information current? do you really have users and logins?
do you still have something that you are calling a template? or are these saved schedules?

design doc
I think that architecture diagram includes uniplemented features. identify. also need to clarify what is deployed on which device
looking for description and interfaces for all of the modules in your architecture diagram
user manuals
not sure that I see how to build the options page
suggest that you put something in the document to point them to free photo editing software and give them a recommended size for photos. They clearly don't need to go very hi-res
test plan
which browsers (s) will you be testing on?
are you really testing the numbers that you say? I thought that you were going to limit the number of pages that could be linked (need to incorporate that in the other docs as well)

USER STORIES:

First time:

Nancy works at an Autism center for adults. She knows that there was a scheduling application that just finished development. So this is the week where she is switching over to the new tool. Nancy opens up the website and starts to make the schedule. Nancy sees three large buttons labeled “Create New Schedule”, “Manage Schedules” and “Upload a Picture”. She clicks “Create New Schedule”.

This brings Nancy to a page with a number of options related to the schdule on it. It has a search bar, to search for images by tag, it has the option of making a “Schedule” or a “Task

Selection", as well as the number of pictures for the page and a space to title the schedule and to assign it to a user. Reading the note at the top, she decides on three images for her Schedule, and then searches for the first tag.

This brings up a group of pictures. Titling the schedule and assigning it to Bob, she hits submit, sending the schedule to the server. She then is taken to what she recognizes as the "Manage Schedules" section, where she is offered the ability to search by user or by title. She chooses Bob from the user list, and this brings up all of Bob's schedules. She can choose which one to make active and which one to appear after the active schedule. She also can delete schedules from here if something went wrong. She hits the "Go Home" button, and clicks upload a photo.

Here she is greeted with the familiar "Choose File" button and a line to add tags. She uploads a picture of people washing dishes, with tags, then goes back to main page to make another schedule with that photo. After finishing the second schedule, she sets the first schedule to be the active schedule for Bob and the second to activate after the first one is finished.

LATER USE:

A couple of months later Nancy is still using the iPad application. It is really easy now that she has a more complete database of pictures; she almost never uses the "upload a picture" button. Today, however, it unexpectedly rained, so Nancy needs to go to the manager, and edit today's plan. She goes in, clicks on "Outdoor Morning", a schedule she had made, and clicks delete. It disappears, and prompts her to select a schedule because there is presently no active schedule. Nancy selects "puzzle time" from the list, and then she adds the food run to follow it up. Satisfied with how this works, she clicks submit, and that changes it on the server.

CLIENT USE:

Nathan turns on his iPad, and is presented with a drop down menu where he picks his name and then hits login. It brings up a schedule with a picture of a lawn mower, a rake and a tv. He knows this means he needs to mow the lawn, rake the leaves and then watch tv. After mowing the lawn, he slides the mow lawn photo across, it locks into place and he gets a check mark over it, letting him know it is done. After raking the leaves, Nathan gets a bit confused and tries to drag the TV image over, but it does not move, and Nathan realizes he was supposed to move the rake image. With that done, he moves on to watching TV.

Next he gets a new screen with two columns, and two rows, he selects 2 tasks and then a new schedule page appears with those two tasks on it. He slides the images across when he completes each task.

USE CASES

CLIENT USE CASE

1. Once id confirmed, load personal schedule
    I. Schedule displays no more than four pictographs at a time.
    II. Two halves, one is to-do, the other is complete
        A. Drag and drop from to-do side to complete side when finished, only allow top remaining task to be dragged.
        B. When all three tasks are complete load next schedule page
    III. On option pages (for selecting between different choices) display buttons with pictographs.
        A. When button is clicked image appears at the top

AUTHORING USE CASE

1. Separate icon for admin page
2. Three options
    I. Create New Schedule
        A. Pick Type of page (Schedule or Task Assignment)
        B. Pick number of images
            1. 1-4 for Schedule
            2. 2 or 4 for Task Assignment
        C. Pick images (can search by tag)
        D. Add title to schedule and submit
    II. Manage Schedules
        A. Select user then search it by title
            1. Selected Schedule options
                I. Activate this schedule (sets to current page)
                II. Make this schedule next (sets the page to follow the active page)

III. Delete

B. Upload a Picture

1. Choose a file and add tags

Requirements:

-General

1. Web based iPad application
2. Must work on a standard webkit based browser

- For users:

1. Login once
2. Choose between two activities several times
3. Drag and drop pictures between unfinished and finished sides
4. Load next schedule automatically

-For authors:

1. Way to upload pictures
    I. Database/server to store pictures
    II. Simple interface
2. Ability to make a day’s schedule
3. Have multiple schedules for a single day
4. Easy schedule access through user

Interfaces:

User front page – drop down with usernames

user page – set of three pictures that can be dragged and dropped to the other side

user task assignment page –one or two columns with pictures appearing at the top when selected

Admin front page— three options “create new schedule”, “manage schedules” and “upload a picture”

admin create new schedule – page to make new schedules/task assignment pages

admin manage schedules page – activate or delete schedules

admin upload a picture – add new pictures to the webspace

Unlimited time:

Error conditions – Test for any failure to load properly

Client Specific:

Test environments – Test on iPad for functionality and on browsers for if it can be looked at

Error Conditions – drag and drop issues for draggable elements, multitouch dragging

Author Specific:

Test environments – Would test the tool on all browsers and mobile devices, there would still be a focus on iOS and webkit, but would test all for extra uses.

Error Conditions – test for broken search terms, search terms that don't narrow options, and multiple word search terms

Data-specific – limits on uploading files, overwriting images, linking multiple pages to the schedule, editing cached pages, limits on saving premade schedules, limits on adding images to templates

Actual Test Plan:

Client Specific:

Test Environments – test on iPad, iOS 5 and 6

Error Conditions –Drag and drop getting stuck or causing unforeseen behavior. End of present schedule chain.

Author Specific:

Test Environments – Focus webkit based (safari and chrome), but also test Firefox and Internet Explorer for most functionality

Error Conditions – upload >100 pictures, search terms returning bad links, no links, or everything, overwriting files

Data-specific – Have a multipage (2) linked schedule, test strange image uploads and denying other file types, proper saving of schedules.

Architecture:

Architecture diagram: on site

This diagram represents the data transfers and work flow of the system. The two main access points are the Client (the one using the schedule) and the Admin (the one making the schedule). The client is capable of fetching a schedule from the database, and storing it locally, and changing it locally (only in order to complete tasks). The Admin can upload pictures with tags and can create new schedules and push them to the database..

In order to ensure this program continues to function, everything needs to be moved and implemented on a more permanent server. It is presently located on a school server and thus may be removed in future.

Module definitions:

jQuery UI – Two key methods: draggable and droppable

Draggable: This gives the ability to drag and move an element (div, header, etc.) with the cursor around the page. We are using the snap and revert attributes.

The snap attribute makes the draggable element jump to the boundaries of another specified element. It can specify the snap tolerance (distance snap will enable) and what element to snap to.

The revert attribute sends the dragged element back to its original position if it is not accepted by any droppable element.

Droppable: This gives elements the ability to be a target for a draggable element. We are using the accept attribute.

The accept attribute specifies which draggable element, once dropped, will activate a trigger on the droppable element.

jQuery – A framework placed on top of javascript. This makes javascript easier to write and code with.

jQuery UI Touch Punch – A script imported into the html document. This forces all cursor and keyboard events to work with touch based input on touch enabled devices.

Processes:

Not relevant

Data:

Actual Storage: There is a folder called uploads where all of the uploaded pictures will be loaded to. The pictures all have unique names and should not be changed. Currently JPEG files are allowed to be uploaded, however that can be easily extended. These pictures stay full size from upload so it will be important to monitor the size of the uploads folder.

Databasing: All of the generation of schedules is done by linking the pictures to different tables. There is a table of images (tbl_images), a table of tags (tbl_tags), and a table of schedules (tbl_sched). The table of images and tags are inherently linked by being added at the same time. The actual current schedule on the client side is pulled down from the table of schedules in a queue like system where there is an ordering and the immediate next is waiting. On the authoring page images and tags are joined tables by their ID. Then the three pictures that are selected make a new entry into the table of schedules. This also puts the new entry in line to be used by the current schedule client side.

In general: The tags and picture tables are joined, and the table of schedules contains three different picture names. The picture names are the same as the actual file names so having the string of a picture name lets you access the picture from uploads.

Design Decisions:

This is the functional spec: on site

The photos are not pushed into the database directly, they are given a unique ID and that ID is associated with the tags. The ID is then related to the photo location. This removes the need to convert the image into a binary large object. The layout of client-side is based on the physical implementation that is already in use. The admin-side is based on ease-of-use, and ease-of-understanding.

Our program is dependent on jQuery, jQuery UI, Touch Punch and MySQL. If any of these suddenly cease to function, our program will also stop working.

We did not manage to implement local admin editing. HTML 5 supports local storage features, and we were looking into that, but needed to focus on other things.