# 6 Step Methodology

# 

1. Rendered Example
   1. A previously done task is show with answer to show user what is expected at the end
2. Guided example
   1. A guided tour of task is show to user how a task is done
3. Feedback
   1. User is showing a task, but option is given to see what is answer to let user know what is expected
4. Evaluation
   1. User is given a task and is prompted weather answer is correct or not
5. Real Task
6. Verification
   1. Already done task is shown with answer asking user to confirm whether the answer is correct, if not then user is asked to correct the answer by editing

# Data Base

**BatchData**

id integer not null constraint BatchData\_pk primary key autoincrement,  
data string not null  
hexID string  
Type string

1. Id
2. Data
   1. Description of batch
3. hexID
4. Type
   1. Main type

**Layout**

id INTEGER not null primary key autoincrement,  
data VARCHAR(2000) not null

1. Id
2. data
   1. layout text in label studio format

**Task**

id INTEGER not null primary key autoincrement,  
text VARCHAR(2000) not null,  
layout\_id INTEGER not null references layout,  
groundTruth VARCHAR(2000) not null,  
format\_type int default 0,  
batch\_id int default 0,  
description VARCHAR(2000) default ""

1. ID
2. Text
   1. Task text data (text/url of image)
3. Layout id
   1. Layout id from layout table
4. groundTruth
   1. correct answer for task
5. format Type
   1. set it to 1 for now, previously used for different type of formats (training/testing)
6. batch id
   1. batch id from batch table
7. description
   1. Task description to be show to be show to user while doing task

**Completions**

id INTEGER not null  
 primary key,  
user\_id INTEGER not null  
 references user,  
task\_id INTEGER not null  
 references task,  
data VARCHAR(2000) not null,  
completed\_at BIGINT,  
batch\_id int,  
was\_skipped boolean

1. Id
2. User id
   1. Id of user from user table
3. Task Id
   1. Task id from task table
4. Data
   1. Task completion data json text, answer given by user in json format stored as text
5. Completed at
   1. Completion time
6. Batch Id
   1. Batch id from batch table
7. Was skipped
   1. A flag if task was skipped by user

**User**

id INTEGER not null primary key,  
name VARCHAR(100),  
username VARCHAR(100),  
type VARCHAR(40),  
password VARCHAR(200),  
created\_on DATETIME,  
last\_login DATETIME,  
is\_admin boolean default False,  
workerId varchar(100)

1. ID
2. Name
3. Username
4. Password
5. Created on
6. Last login
7. Is admin
8. Worker Id
   1. MTurk worker id/ google ad cookies-based session id of user

**UserScore**

id INTEGER not null  
 constraint user\_score\_pk  
 primary key autoincrement,  
user\_id int,  
batch\_id int,  
score float,  
showDemo Boolean default TRUE,  
current\_task\_type int not null

1. Id
2. User id
   1. User if from user table
3. Batch id
   1. Batch id from batch table
4. Score
   1. User score for a batch, incremented after each completion
5. showDemo
   1. used this flag initially to keep track if we need show demo to user based on his score
6. current task type
   1. keep track of current step of 6 step methodology

# Flask module / Backend

Main landing page for User

@blueprint.route('/<batchid>')

def labeling\_page(batchid = '0'):

params: Hex batch\_id

from Mtruk:

worker\_id, hitId, turkSubmitTo, assignmentId, gameid

Get numeric batch\_id from DB corresponding to hex batch\_id

Check if user is coming from Mtruk / google ads/ used login screen

For MTurk check if worker\_id exist in params

For login check if flask\_login.current\_user exist

For google Ads maintain cookies-based session

For New User create UserScore entry

Get Next task for given user and batch\_id

Convert to labeling studio Json format

@blueprint.route('/AdminLabeling')

def admin\_labeling\_page():

params: Hex batch\_id

Allow Admin user to do tasks to add to system as examples for other usere

Show other User’s completions for tasks to make add to system as exampls

@blueprint.route('/noAuth')

def not\_authorised\_page():

@blueprint.route('/noAuth')  
def not\_authorised\_page():

@blueprint.route('/batches')

def batches\_page():

Show all batch added in system

@blueprint.route('/importTasks')  
def import\_Task\_page():

Import task page for our need, support only CVS

Allow admin to import task and completion data for a batch

Allow admin to import only Tasks ( new tasks)

Allow admin to import new Layouts

Uses following URL to upload data

@blueprint.route('/api/project/importTasks', methods=['POST'])  
def api\_task\_import():

params: UploadType (tasks/layout/twc (task with completions))

CSV file

@blueprint.route('/api/project/next/<batchid>/', methods=['GET'])  
def api\_generate\_next\_task(batchid):

Api URL used to generate next for user after completion of a task

Use same params as labeling page

If user is admin then check if there are any tasks completed by other users and mark them as new completion by admin, if not then show next task available

@blueprint.route('/api/tasks', methods=['GET', 'DELETE'])

def api\_all\_tasks():

params: batchid

Show all tasks done by user

@blueprint.route('/api/tasks/<task\_id>/completions', methods=['POST', 'DELETE'])  
def api\_tasks\_completions(task\_id):

Save new completion

Params: same as labeling\_page to identify from where ths user is coming (Mturk/login/google ads (cookies)

was\_cancelled : was tasked skipped

completion: json data

Save users response in db for task then decides to move forward for next task type

If current task type is

Need 1 completion to move from step 1 to to 2 and from step 2 to 3

At step 3 number of completions required to be completed is X (for now it’s 2)

At step 4 number of completions required to be completed is X (for now it’s 2)

At step 5 and 6 a uniform random number is generated

If it is between 0 and 0.2 then set next step to 6 otherwise keep set step to 5