This document acts as a starting place for all of the Truman documentation. Please start and refer back to this page for help in using the Truman Platform. This is currently a living document that will continue to grow and change. Once this documentation becomes more stable, this and the rest of the Truman documentation will be moved to a more permanent location like a GitHub wiki or website.

As you read this page (and other pages in the Truman documentation), feel free to add comments and questions.

The Truman Platform

What is the Truman Platform?

Named after the 1998 film, *The Truman Show*, the Truman Platform is a social media simulation platform created by The Cornell Social Media Lab with the goal of providing social media researchers a place to conduct their experiments in a more ecologically valid environment. Truman gives researchers lab-like control over study conditions in a more realistic social media setting. The Truman platform looks and feels like a real social networking site, because it is one. Researchers can curate, create, and control every user, post, like, reply, notification, and interaction on the social networking site. To do this, the Truman platform manages parallel simulations for all study participants. When the participant makes a post, the bots (called actors in the Truman Platform) are pre-programmed on how to reply back. These actors also read and like the participant's posts, creating new notifications for the participant. Therefore, all participants will be exposed to the same social interactions, posts, and responses within a controlled environment that looks and feels realistic. They don't connect or interact with any other real participant on the site, even though they believe they do. All of this is done to ensure participants have an identical, naturalistic social media experience, except for variations controlled by the experimental condition of the study and participant's own posting behavior, which are variables we want to examine.

We believe the Truman Platform could be a great tool for many other social media researchers because of its ability to fabricate real-life online social interactions. In past studies, we have built a social networking site (SNS) called EatSnap.Love on top of Truman where people can share, like, and react to pictures of food. With features similar to popular SNS such as Instagram and Twitter, EatSnap.Love has been an incredibly useful space for us to develop, implement, and test our designs and interventions in a practical, dynamic manner. As we continue to use the Truman Platform to design new studies, we would like to share this platform with other researchers so that we can create new knowledge together.

Truman is freely available on GitHub (https://github.com/cornellsml/truman) for complete replication of any study and creation of new experiments, and runs on any web based platform. While Truman is not unique in creating a social networking lab with bot-controlled users, it stands out by offering a complete experimental platform for managing participant simulations, surveys, data collection, participant observation, experimental condition pool assignment, and

participation reminders, bringing new opportunities for researchers without technical skills to design, implement, and run their own experiments.

DESIGN FOR THE STUDY

Why Truman?

Truman isn't just another tool to simply collect data. It's more than that. Not only can it collect psychometrics, such as tell you how participants are feeling or give you an insight to their thoughts, but Truman produces an actual social media simulation. Unlike surveys, screenshots, and vignettes, Truman allows for the collection and observation of actual behavior. As such, Truman works best in experimental designs that wish to capture behavioral data along with psychometrics.

Truman shapes an environment to mimic a social media platform to capture online behavior in the form of behavioral metrics, like commenting on a post, flagging a message, or navigating to a page. Truman can also keep track of the amount of times a participant has logged in and how many posts they've made and viewed per day. However, Truman will not be able to obtain any metrics outside of the website or browser (like geo location or facial expressions)

By using Truman, participants will not know that their behavior is being monitored, but provides no risk to participants as it simulates the usage of social media. Furthermore, Truman normally uses Amazon's MTurk system as a recruitment pool, so personal information of participants are not matched to actual responses in comments, nor actual photos uploaded for their profile picture or that of posts during the study, to ensure the safety and anonymity of the participants.

What Does a Truman Study Look Like?

A Truman Study allows for observation in a real environment, with a desired behavior as a dependent variable. A truman study can start with a pre-survey to gather information about participants before they begin the study. After the pre-survey, the study is activated and can run for a chosen number of days. Truman assigns participants into experimental conditions and randomizes the conditions and variables. During this time, Truman can automatically track and collect data on desired interactions, such as likes, comments, flags, if the participant checks their notifications, number of posts, time on site, and more. These are stored in a spreadsheet for easy viewing and comparison. Participants receive reminder emails throughout the process to encourage them to meet the criteria of the study. After the allotted amount of time is over, participants lose access to the site and are directed to a post-survey that is connected to their participant identifier. To summarize, each participant is given a pre-survey, a set amount of time to interact with the interface, and a post-survey. Truman provides an easy onboarding process so that participants can continuously be added to the study. Each participant added does not interfere with participants currently in the study.

****Link to papers****

- **CHI 2018 paper** https://cpb-us-e1.wpmucdn.com/blogs.cornell.edu/dist/c/6136/files/2013/12/cyberbullying_truman_final-26r0ofh.pdf
- **CSCW 2019**

LIFE CYCLE

Initial Design

As a researcher, you can manipulate features of Truman to fit the needs of your research. The variable that you manipulate is considered the independent variable (IV). Things that you may change are the looks/ identity of people (bots), the posts of those bots, and the interactions between these bots so that it will fit the variable you are manipulating (or your independent variable). You can also change the interactions of the bots to the participant.

* You may also change the interface of the environment, however, with this version of Truman, only an Instagram-like user interface (UI) is available. Coding experience will be needed to change the UI/UX according to experimental groups

The variables that are affected by the independent variable are also known as dependent variables (DV). On Truman, the DVs are typically the response of people or their resulting behavior from the participants. Depending on your independent variable, depending variables can include what they post, like, comment, or flag. It can also be the time they spent on the website or the frequency on which pages or profiles they click on. Summer of data collected is below

All interactions (posts, comments, likes, flags)
Actor based interactions (Block, report, profile view)
Site logs (time on site, page/notification views, posts views)
Participant information (profile image, name, browser used)

Other considerations

Cover Story
You should think about building a cover story for using Truman. We usually have a cover story about beta testing a new social media application.

Ethics
Getting ethical approval for Truman studies can be a little more tricky, as it deals with deceiving participants into believing all of the actors are in fact real users of the site. A key element to getting ethical approval is to a clear debrief in the post survey at the end of the study, and allowing participants to make an informed consent after this debrief.

Developing Simulation

Installing Truman
Go here (https://docs.google.com/document/d/1pUewmkZUKI1d7lcB0c-VL7eyinyAvC36_Oq7UKo8Hos/edit#) to install Truman on your local machine.

Timeline
Truman follows specific timeline guidelines to ensure your study runs in the way you would like. Each participant has their own individual timeline within the study. This timeline is centered around the moment of each participant's account creation. The study is broken up into 24 hour windows that are constantly moving with physical time, and everything else is relative to that 24 hour window. When you create posts to be on the site, they require a time for the user to see that they have been posted. Anything given a negative time will be viewed as something posted in the past, or before they joined the site, while anything with a positive time will be viewed as

something posted in the future, or after they join the site. These posts should be staggered throughout the duration of the study (with some negative times) to emulate a real experience.

Creating a Simulation Script: How to translate your design into simulations

You can begin with the default basic script that the Truman ships with, but a specific simulation with different conditions will require its own custom script.

Here's what you need to build your own script:

- Actors
    - You'll need to create personas for all actors in the simulation. This includes information such as user names, names, and profile photos.
- Posts and notifications
    - All posts will include images and/or text. You'll also need to consider the timing of the posts, and types of notifications that users receive.
- Experimental message
    - What is the goal of your lesson? (Cyberbullying? Fake news?)

      Consider annotating data: You'll likely want to annotate any subset of data you want to target. Annotating certain posts (e.g. bullied posts v. normal posts) differently will make it easier to isolate the data you're interested in, and look back at your target data after the experiment.

After creating the piloting and testing your simulation, you can push any new changes to Git and continue to refine your study.

See https://docs.google.com/document/d/1GHOmo3c-t7Xhzb53DsQdd52zJ73x68fwch9-yn7dvss/edit
For more details on creating the simulation.

Piloting and testing

Hosting Truman
To pilot and test Truman with other people (outside of your laptop) requires creating a public facing version of Truman. Creating a publicly accessible instance of Truman requires some server skills. There are many many ways to run a public facing Node application (of which Truman is). You will need to host it so that other people can use the site.

Here are some guides in hosting and deploying Truman

- https://github.com/sahat/hackathon-starter#deployment - this has a few quick guides on hosting a node application on a number of different services like heroku, Microsoft Azure, and Google Cloud.
- Another great tutorial I used for Digital Ocean is here (https://lengstorf.com/code/deploy-nodejs-ssl-digitalocean/)

- Some more digital ocean tutorials
    - https://www.digitalocean.com/community/tutorials/how-to-set-up-a-node-js-application-for-production-on-ubuntu-16-04
    - https://zocada.com/deploying-node-js-application-digitalocean-setting-server/
- For Amazon Web Services see https://aws.amazon.com/getting-started/projects/deploy-nodejs-web-app/

To send email from Truman you will need to get a MailGun API keys. Go to (https://github.com/sahat/hackathon-starter#obtaining-api-keys) and look for mailgun for more instructions.

For a lot a great insight to running a production version of node, read the README at https://github.com/sahat/hackathon-starter. Truman came from the skeleton of this codebase.

Other things to change

- Logo of the site
    - Create a new logo called “logo.png” and place it in the “public”’ directory
- Name of the site
    - Change eatsnap.love into your site name throughout the pages in the “view” directory
- In .env change “POST_SURVEY” to the link of your post survey
- Create new New Onboarding rules/pages in “View” directory
- Change CSS in /view/ui_layout.pug to change the look and feel of site

Testing

Before running your actual study, you can test out your study yourself (and with other researchers) to finalize the design of the study. Some things you want to look out for are broken functions and grammar mistakes. To effectively make use testing, try to test every single condition (variable) in your study. Make sure that the condition is actually displaying the condition you want. Also, while Truman does automatically manage the time scope of the study, it won’t hurt to double check that the simulation ends when it is supposed to end and also whether the emails had promptly sent automatically.

Best Practices:

- Create document that covers all user stories and interactions (everything a participant can actually do on the site)
- Have at least two team members check and sign off on each study condition ensuring complete consistency with that study condition
- Run thru whole experiment (all days) multiple times, reading the posts, reading user profiles, ensure consistency in stories of each actor
- Ensure all user capabilities work (upload, comment, view, read)
- Ensure data capture is working and collecting exactly what you want

Piloting

In addition to testing, it's good to run a few pilot studies. Pilot studies are a condensed version of the actual study. They're meant to test out and sort kinks before running the actual study with participants. This will help guarantee that the actual study can run as smoothly as possible. Additionally, the results of the pilot study can give you a taste of what the actual study will produce, and can steer you towards making adjustments on your current study design. The more you test the design of the study in the beginning, the less you have to worry about problems down the line.

Pilot with both participants that you can meet and interview (so you can get more information about the experience of the experiment) and with the actual participation pool (MTurk) so you can test all the parts of the study on a small sample before you launch for a full study.

Running the Study: Server Building and Participant Recruitment

Once you are satisfied with your simulation, it's time to run the study.

Recruiting Participants

The easiest way to recruit research participants is through the use of crowdsourcing platforms. Amazon's Mechanical Turk (MTurk) is one such platform. Your study can be posted as a "HIT" (Human Intelligence Task) on MTurk's database, giving you access to tens of thousands of possible participants. Payment for participants can be set up in a number of ways, like payment at the time of total completion or payment for total number of hours/ days spent on the Truman instance.

Running a Mturk study

For more information about mturk (https://docs.google.com/document/d/1x9SxmfrAeniDJUlrYEqUq3bkfZRGp1t6k7AkBUE3pqs/edit)

See (https://docs.google.com/document/d/18MbFjZAebWiv6kEXvcDJNxpyy45EIcINhfJMkFdCwjc/edit) for some more information on running a mturk study

See (https://docs.google.com/document/d/1EQ9BjvvVd_SsOA8dogXFVbND-AAP35irBH11EykKCUc/edit) about paying mturk participants.

Collecting and Analyze Data

At the end of your study, you should have three sources of data: The pre-survey, Truman site data, and post-survey. The Truman site data is in the MongoDB database you set up.

To export this data into a csv file, run node data-export.js

Collecting Data

By using the data-export.js file, you can create a number of csv files that tailors to the needs of your research. An example is a summary file, where you can get a complete set of data that compiles post-study behaviors for you to analyze. You may gather global variables, such as how many times the participant has logged in, the amount of posts they liked, and other summary descriptives. If needed, you may collect more in depth metrics rather than just the surface level summary. All user collection information will be stored in the Mongo DB, however this can be a bit overwhelming to look at. The csv files are more condensed versions of the entire information. You may need to modify data-export.js for your own data needs.

You will also need this data to help pay your participant on MTurk, as you will need to know how far each participant made it through the study, and how much you owe each participant.

Analyzing Data

When analyzing data, you might be interested in comparing different conditions. You may use programs like R, or other analyzing tools for finding interesting trends and statistical significance in your data.

The key that unites the participant data files is the Mechanical Turk ID For the pre-survey, the behavioral data from actual study, and the post survey. You may also use the Mechanical Turk ID to identify and help with payments/ rewards after the study.

Truman4Researchers

Manual for Installing Truman

Introduction:

Hello Researcher!

Thank you for choosing Truman for running your studies!
However, before we can get Truman up and running, we would need to prepare your local computer to run Truman.

In this document, you will be preparing your local device with all the necessary prerequisite software for running Truman.

Truman is a web application that uses the Node.js framework. This means you must first install Node.js and a version of Git to download and run the Truman source code. These instructions will help walk you through this.

Brief Overview:

This is a brief outline of what this document will instruct you to do:

1. Installing prerequisite software
    a. To run Truman, you must first install node.js on your local machine.
    b. To do this (and to complete other steps in running Truman), you will be using the Terminal (**for Mac**) or the Command Prompt (**for Windows**)
    c. Terminal/Command Prompt:
        i. It is an application that provides text-based (commands) access to one's operating system (computer)
        ii. It can be much faster to complete some tasks than with graphical applications and menus.
        iii. Another benefit is allowing access to many more commands and scripts.
            1. On Mac, click the search "magnifying glass" icon at the top right of your desktop, and type "terminal" into it (or use shortcut command + spacebar to open up search)
            2. On Windows, Press Windows+X to open the Power Users menu, and then click "Command Prompt" or "Command Prompt (Admin)."
2. Creating An Account on MongoDB Atlas
    a. MongoDB is the database system that Truman uses. MongoDB Atlas is a cloud based MongoDB service that allows us to easily set up a free MongoDB Database.
3. Create a Github Account and Create your own version of Truman
    a. All of the Truman code is hosted on GitHub, you will need to connect to GitHub to download and use the code
    b. When you make your own project, you will want to do this with your own copy of Truman, hosted on your own repository. This allows you to easily save and share your project anywhere.
4. Cloning (downloading) the your version of Truman from Github to your computer
5. Find and Change Local Files
6. Populate your Database
7. Using Truman for the First Time

Instructions:

Step 1: Installing Prerequisites

A. (Mac Only) Install Homebrew (http://brew.sh)
   a. A free and open-source software package management system
   b. Simplifies the installation of software on Apple's macOS
      i. You can install the rest of the prerequisites with Homebrew (reference below)
   c. This **does not** work for Windows

B. Install Node (https://nodejs.org/en/)
   a. Platform that uses Chrome's JavaScript Engine to run code on a server instead of the browser.
   b. Helps to easily and quickly build web applications on the server-side (back end)
   c. Without Node, the Truman code would not run
   d. (**For Mac**)In the Terminal, type the following to install Node: brew install node
   e. (**For Windows**) Use these instructions to install node on Windows (https://www.guru99.com/download-install-node-js.html)
      i. You don't need to install chocolatey if you follow the installation wizard discussed at the start of these directions.

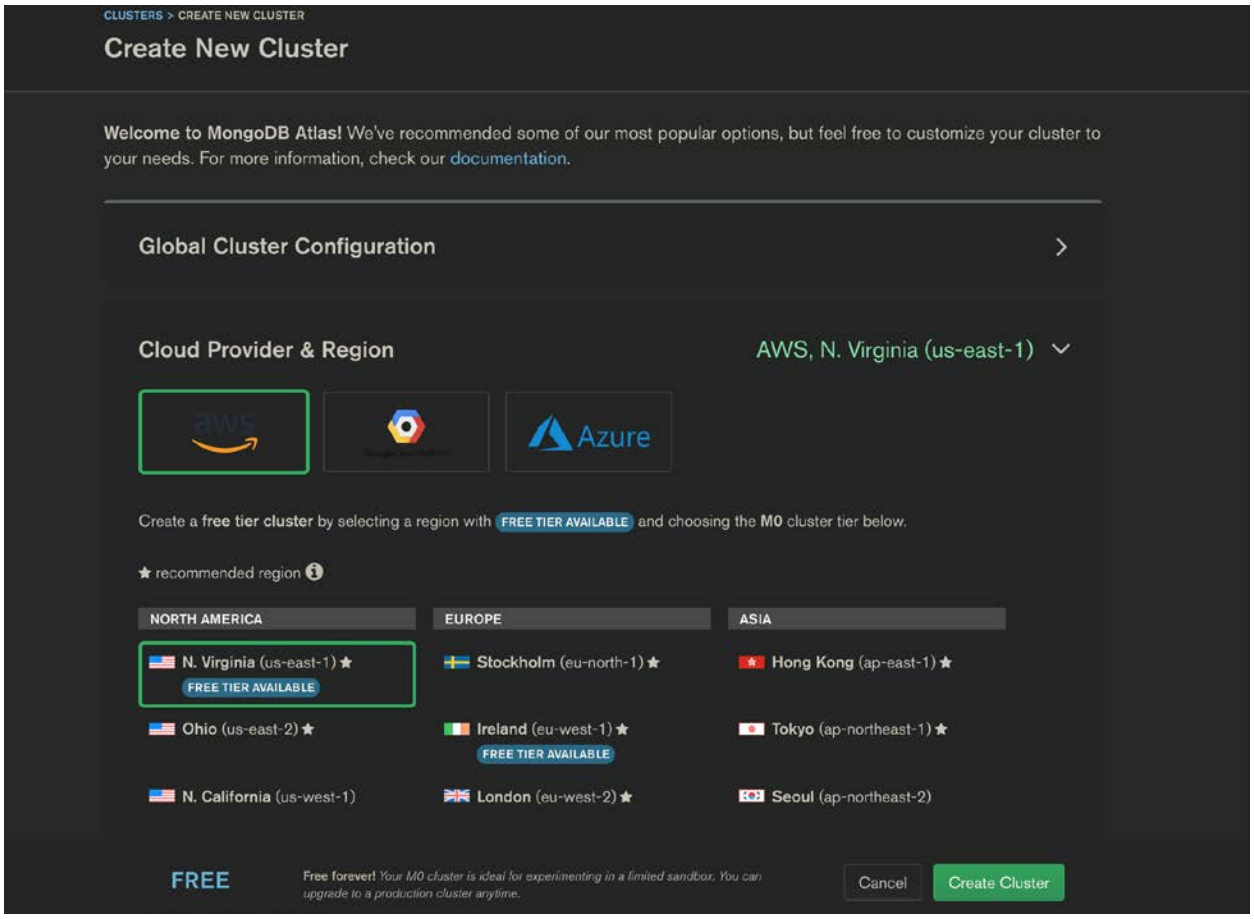
C. Install Github Desktop (https://desktop.github.com/)
   a. A version control system
   b. Tracks changes in computer files
   c. Files (code) can be maintained amongst multiple people
   d. Go through these instructions to download and install it (https://help.github.com/en/desktop/getting-started-with-github-desktop/installing-github-desktop)
      i. If you don't have a Github account, be sure to sign up for one as well

D. Verify that all prerequisites are correctly installed
   a. To check the version of the prerequisite (Make sure you have Node 8 or higher):
      i. Type into the Terminal (**for Mac**) or Command Prompt (**for Windows**), the following:
         1. For Node: node --version
         2. It should return "v8.0.0" or a number higher

Step 2: Creating an Account on MongoDB Atlas

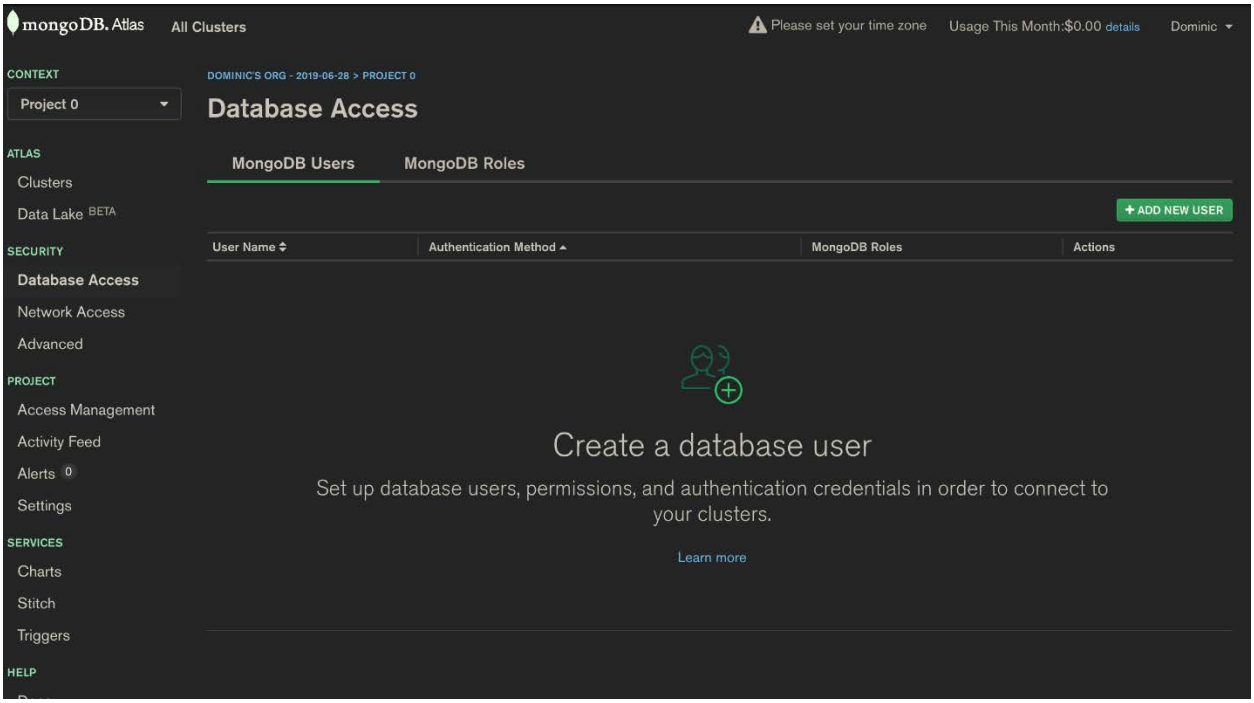
MongoDB Atlas is a managed cloud database service that hosts MongoDB databases

A. Make a database
   a. Navigate to https://www.mongodb.com/cloud/atlas
   b. Click the green **Get started free** button
   c. Fill in you new account information and then click **Get started free** button
   d. You will be directed to the **Create New Cluster** page.
      i. Make sure AWS is selected
      ii. Choose one of the “Free Tier Available” regions. Like N. Virginia
      iii. Give your Cluster a name (the default is Cluster0)
      iv. Don’t bother changing any of the other selected default options
      v. Click on green **Create Cluster** button



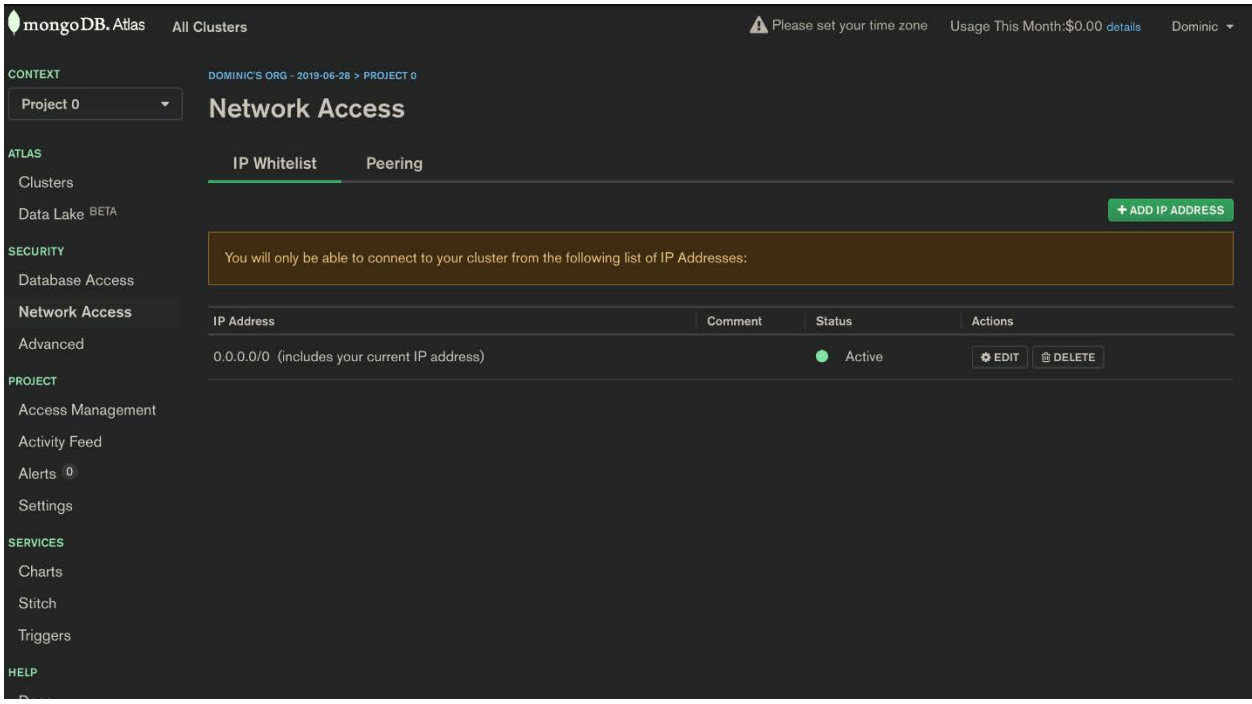
   e. Wait for a while as your cluster is created (might take 5-8 minutes)

f. Now that your cluster is ready, we need to create a DB user. To create a new MongoDB user, from the Clusters view, select the **Database Access** page found in the Security tab on the left.

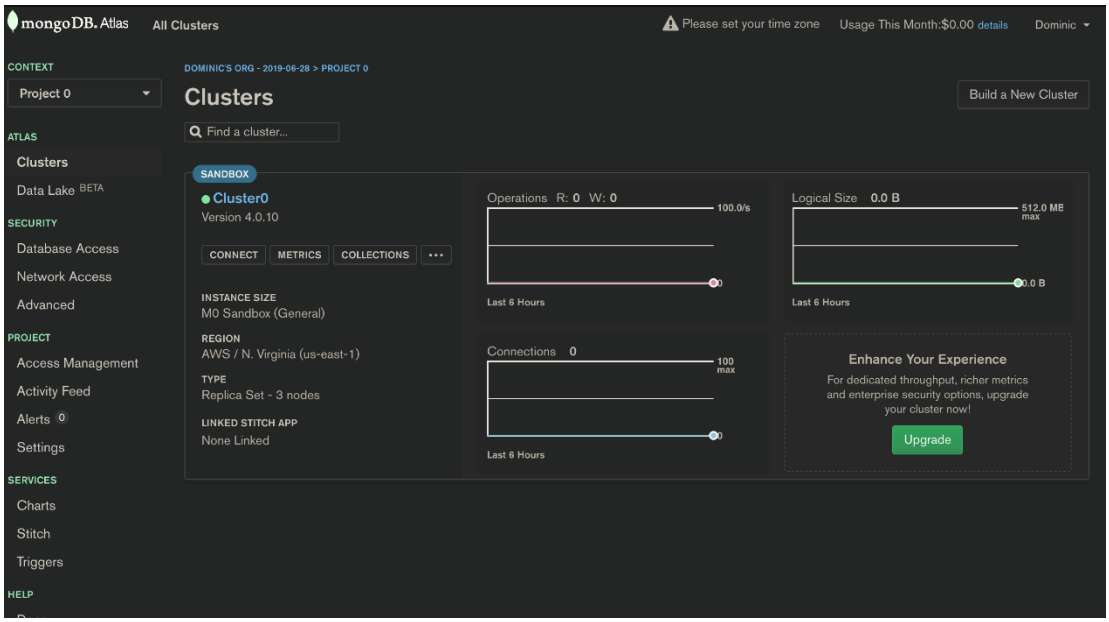


g. Under the MongoDB Users tab, click on **+ Add New User** button
h. Fill in a username and password and give it Atlas Admin User Privilege
i. Next, in the Clusters view, click on the **Network Access** page under the Security tab. You need to create an IP address whitelist and obtain the connection URI.
j. Click on the green "**Add IP Address**" button at the top right.
k. Under section (1) Check the IP Whitelist, click on ALLOW ACCESS FROM ANYWHERE. The form will add a field with 0.0.0.0/0. Click Conform to save the

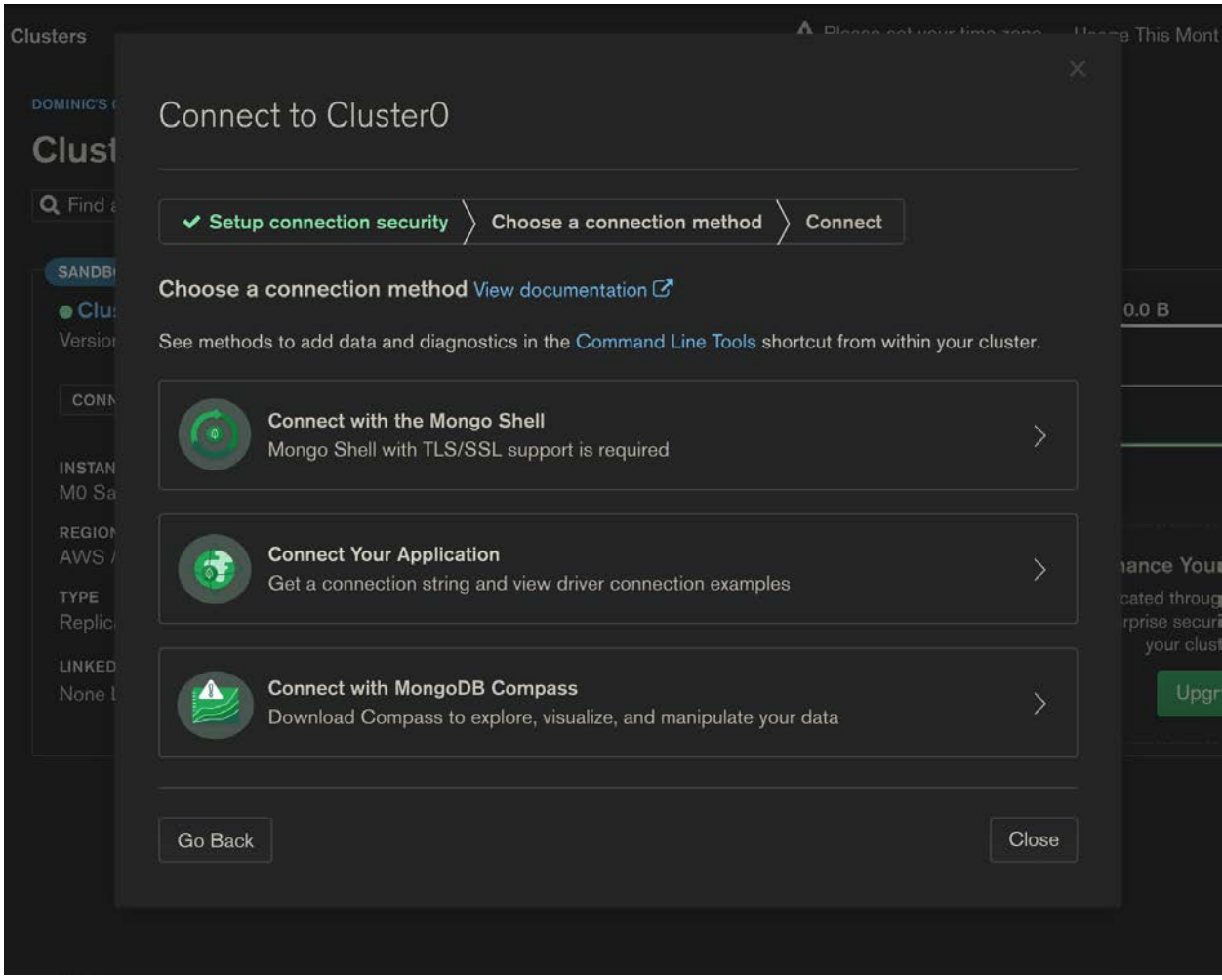
0.0.0.0/0 whitelist.



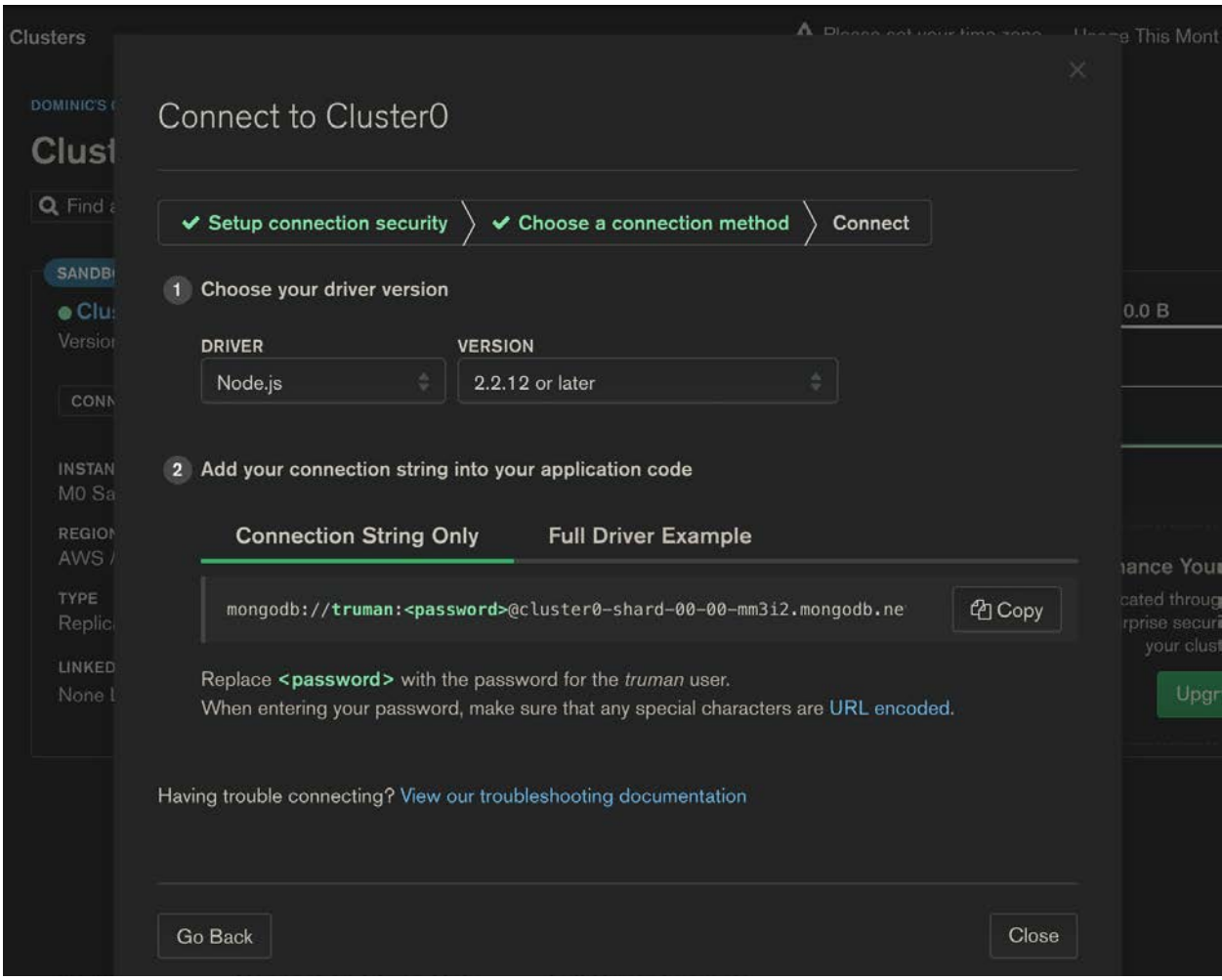
l. Go back to the Clusters page under the ATLAS tab and click on the **connect** button under the name of your cluster (Default was Cluster0)



m. Select "**Connect Your Application**" in the follow window.



n. In the new screen, select Node.js as Driver and version 2.2.12 or later. ***WARNING***: Do not pick 3.0 or later since connect-mongo can't handle mongodb+srv:// connection strings.



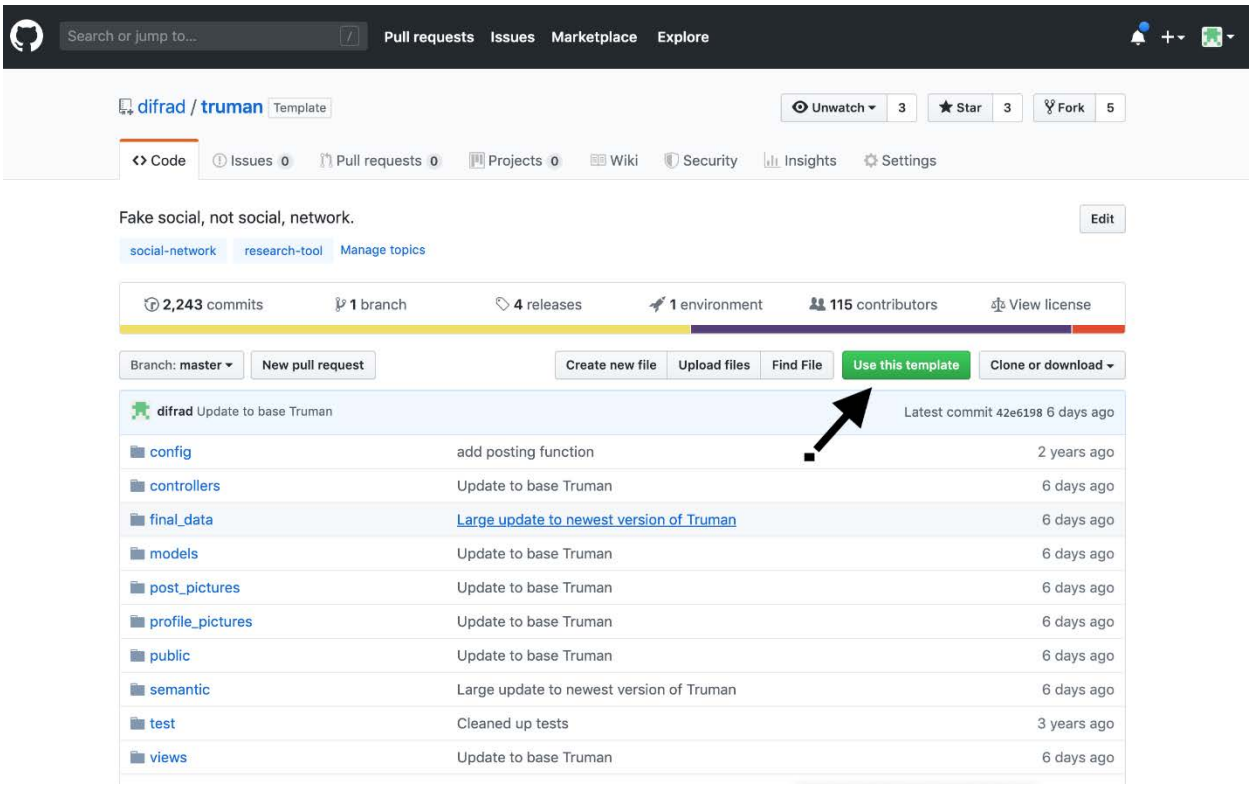
o. Finally, copy the URL Connection String and replace the password section with the password you used for the Database user you just created (hint: don't include the <> when you replace the <password>. Only your password should be in this part of the URL.) Keep this URL handy as we will use it later to connect Truman to this database.

Step 3: Create a Github Account and use Truman Template

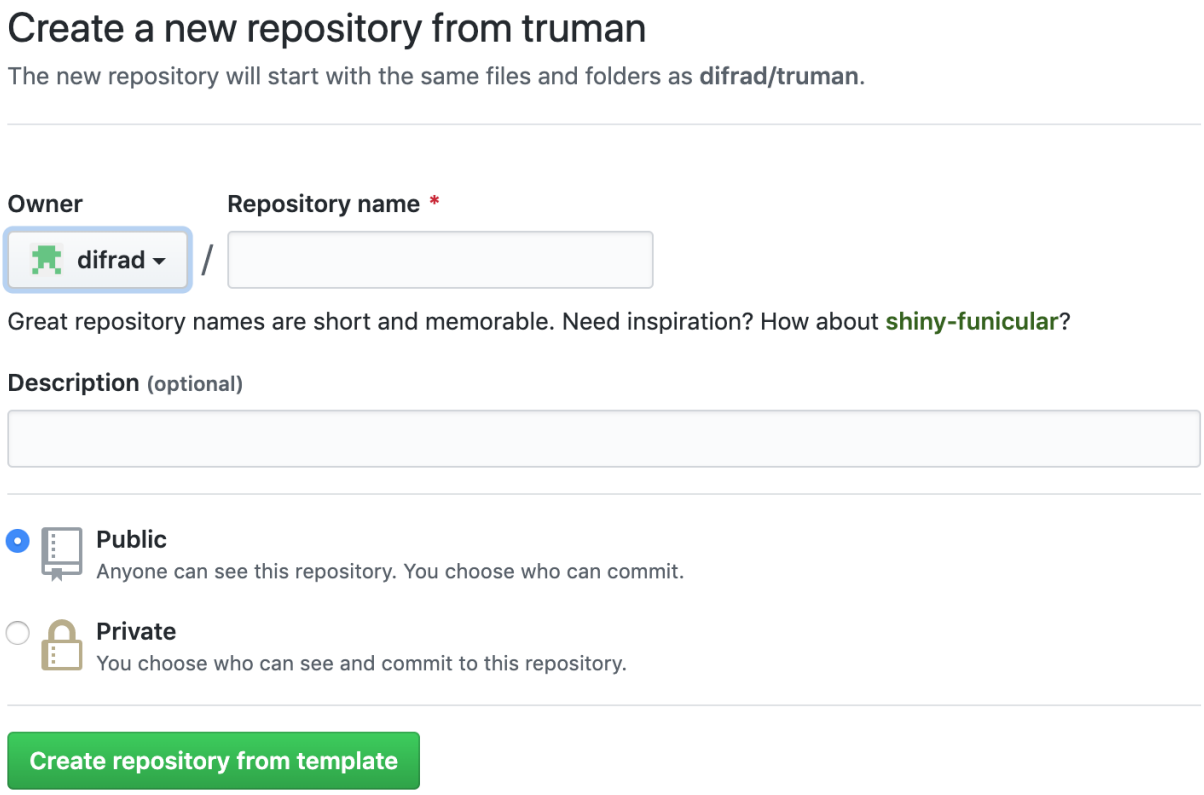
Next we are going to create a Github account (if you don't already have one) and use the Truman code template to create our own code repository of Truman. This allows us to create our own projects without affecting the main Truman source code.

Sign in to Github, go to https://github.com/cornellsml/truman

At the top you will see a green button that says “Use This Template”.



Click on the green “**Use this template**” button and start the process of creating your own repository. Give your repository a name, and make sure that the Public setting is set.



You now have your own version of Truman on your own Github account.

Step 4: Cloning the Truman Project from Github

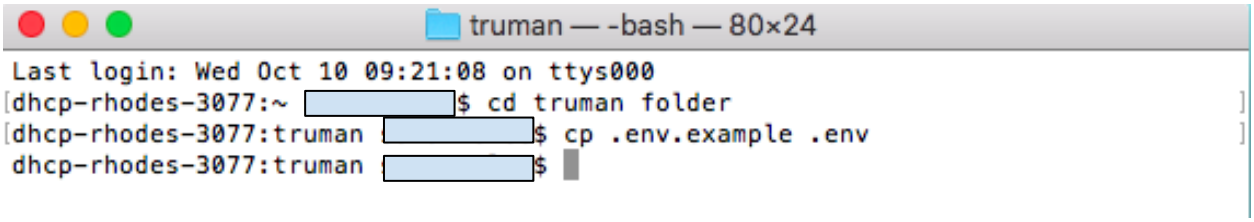
Now that we have a version of Truman in our Github account, lets clone this codebase on our local machine. Cloning creates a local copy of the code on your own machine. When you make changes to the code on your local machine, you can "push" these changes back up to your remote repository that's on Github.

For instructions on how to clone your repository, see (https://help.github.com/en/articles/cloning-a-repository) You can either use the terminal (as seen in the above instructions) or use the Github Desktop application (see here for instructions on how https://help.github.com/en/desktop/contributing-to-projects/cloning-a-repository-from-github-to-github-desktop)

Be sure to remember where you clone the repository.

Now, open the terminal (for Mac) or the Command Prompt (for Windows)

A. cd (change directory) into the directory you cloned the repository into. For example if you named the directory “truman” you would enter cd truman
B. Next, type ⇒ npm install into the command line to install all the external node libraries for the truman project. You are going to see a bunch of lines of code come and go on the screen. This is completely normal. This may take a few minutes as the project needs to download and install a lot of libraries.



Step 4: Find and Change Local Files

A. Create .env and edit
   a. The purpose of the .env (environment) variable is such that the researcher can locally store their MongoDB passwords on the machines rather than on the cloud server where the passwords would be visible to the public.
   b. You'll be copying the example environment document we've provided and modifying it by renaming the usernames and passwords you've created in MongoDB Atlas [this is where you'll refer to the text file where you saved your URL with your account credentials mongodb://<dbuser>:<password>@.....].
      i. This is where you will be replacing <dbuser> and <password> with the **database user account** information that you just created. (e.g. mongodb://**myusername**:**mypassword@**.....)
      ii. There is a file named .env.example, containing environment variables set with default keys. We'll be copying the .env.example file and creating a local .env file, and modifying it to contain your specific keys. This is where you'll use the MONGODB URI created earlier through MongoDB Atlas.
   c. env.example (found in the github project) are example keys.
      i. Type cp .env.example .env in terminal (Mac) or copy .env.example .env in command prompt (Windows)
         1. Copies content in .env.example and names the newly copied file **.env**
   d. Modify mongo URI in new .env (MONGODB_URI=)
      i. Replace the current key with your MongoDB URI from MongoDB Atlas:
      ii. You can do this through a text editor if you can't find the file through finder (.env is a hidden file)
      iii. (**for Mac)**
         1. Edit with nano by typing this in the terminal (nano .env)
         2. After "MONGODB_URI=" add in your MongoDB URI from MongoDB Atlas
            a. Do the same for "MONGOLAB_URI="
            b. Don't change the other variables at this time
         3. See screenshot below for an example of the .env file
         4. Finish by exiting the nano window with **CTRL + X**
      iv. (**for Windows**)
         1. Enter .env in the command prompt, this should open it in a text editor.
         2. Update the file as directed above and save the file

Quick Reference of File Structure of the Node Project

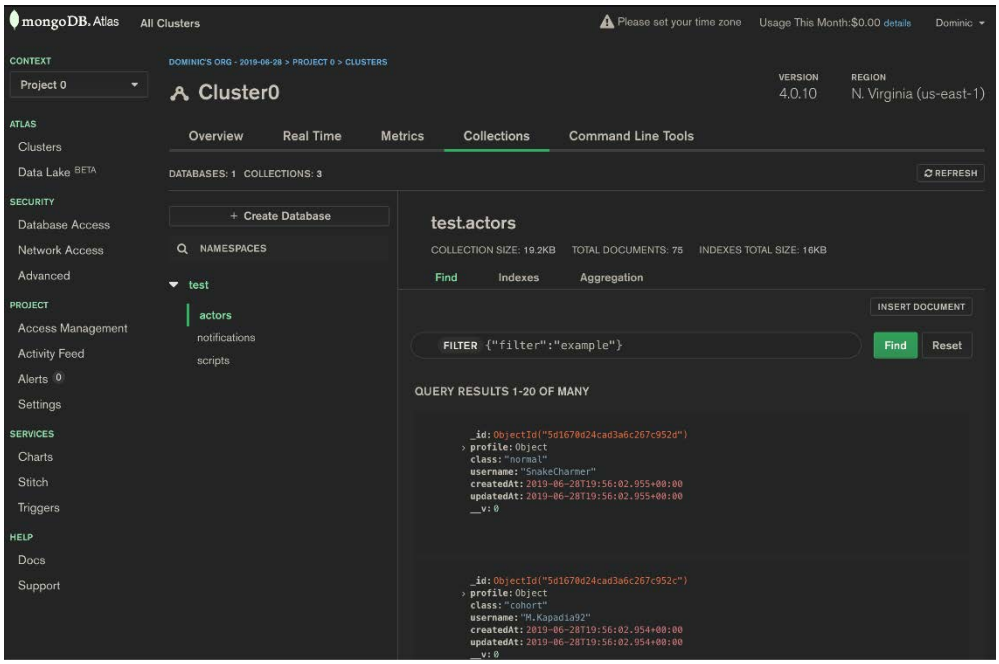
Filename	Description
App.js	Homebase. This is where you kickstart the program (what you run to turn the whole project on)
README.md	Descriptions about the project for more information
Results folder	This is where your custom data would go once you run your own experiment.
post_pictures/profile_photos folders	Default/example images included, but researchers are free to include their own images in this repository
Uploads folder	Where your participants photos will go when they upload to the site
Public folder	Files relating to front-end development (e.g. CSS)

Input folder	Where you place the csv files that will make up the simulation for your project
populate.js	Use this program to upload your simulation (found in the csv files in the input folder) to your new mongoDB database and create your simulation
data-export.js	This will create csv files from your database into the Results folder. Use this to get the data at the end of your study.

Step 5: Populate your Database

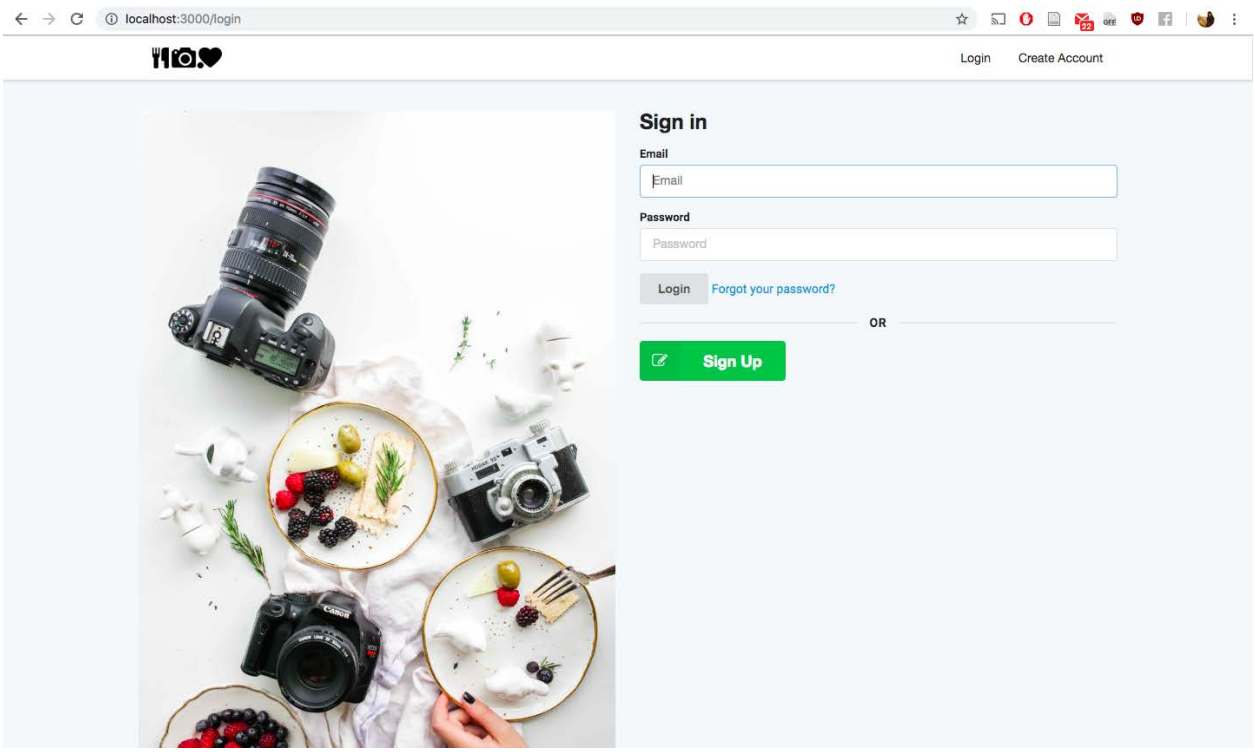
Now that we have set everything up, it's time to populate your MongoDB database

A. Type node populate.js in terminal/command prompt
   a. This will connenct to the MongoDB database you just created and upload the simulation data found in the csv files in the Input direction.
   b. Go to cloud.mongodb.com again, and look at the database you created. You can see it in the **Collections** button in the cluster you first created. You should see 3 new collections in your database called “actors”,
   c. “notifications” and “scripts” (see screenshot for an example)



Step 6: Using Truman for the first time

A. Now that everything is installed, we can now run Truman on your local device! Type npm run dev in terminal/command prompt
B. Type localhost:3000 in your browser of choice
C. Here you can create a new account and try out Truman for yourself



Truman Simulation

What is a Simulation?

Each individual Truman simulation is based around the moment the user joins the site. T_0 when they joined. Every post on the site is in reference to this point in time. For example, -12:30 is 12 minutes and 30 seconds before they joined, and 62:31 is 62 minutes and 31 seconds after they joined. Keep in mind that this is unique for each account that is created--there is no set time that a simulation begins or ends. The way simulations are creates is twofold; each one is built with Actors and Posts. Each component of the simulation is described below.

1. **Actors:**
    a. Within the simulation, actors are the users of the system itself. To create an actor, you will need information like a profile picture and username.
    b. The folders containing the information/files you will need for the actors:
        i. Profile_pictures
            1. The smaller the better
            2. Using a compression software that compresses the image and removes any metadata is encouraged.
                a. ImageOptim is one example.
        ii. Bots.csv
            1. Different attributes for the user
            2. *Username is important!*
                a. Unique ID *has* to be unique.
            3. The picture column references the name of a file inside the profile_picture folder. When filling this in, it *has* to be exactly the same as it is in the profile_picture folder.
            4. The class column can be used for experimental purposes.
                a. For example, you can label the actors as either “bully” or “victim”
                b. This can also be left blank.
    c. The *Bots.csv* file is where you will fill in the information for the actor. The *Profile_pictures* folder is where you will find available photos to use in your profiles. When you are choosing a photo from the *Profile_pictures* folder, make sure the name matches **exactly** what is being entered into the *Bots.csv* file, or else it will not work!
2. **Posts:**
    a. Associated with each actor are *posts*. You will need a photo and a caption for each one.
        i. The folder containing the images used to create the posts is:
            1. Post_pictures
        ii. Post.csv
            1. This is where you will create the posts for all of the actors.
            2. The “ID” column has to be unique inside the table. This helps to identify the *post*.
            3. The “Body” is the text under the post itself. This can be just text, but it can also be emoji.
            4. The “Pictures” column has to match what is in post_pictures exactly.

5. The “Actor” column is referencing the username of the actor in the bots.csv. This also has to match exactly.
6. In the “Time” column, you will add a time stamp to each post that is in reference to the moment the user joined the site.
    a. So -12:30 [12]mins [30]seconds before they joined
    b. 62:31 [62]mins and [31]seconds after they joined
7. The “Class” column can again be used for classification for experimental purposes, or left blank.

3. **Actor Reply:**
    a. We want to generate notifications to the users on the site.
        i. The “ID” column is the unique ID for the post itself.
        ii. “userPostID” corresponds to the participant’s post.
            1. 0 is the first post
            2. 1 is the second post
            3. Etc
        iii. “Time” in this case is relative to the user generated post.
        iv. “Class” again, can be used for classification for experimental purposes, or left blank.
4. **Notifications:**
    a. UserPost: corresponds to the post of the participant
    b. Type: the type of action that took place
        i. Most commonly is like.
    c. Actor
    d. Time: relative to the post
5. **To run all of this:**
    a. Make sure all of your csv files go into the input folder
    b. Run node
        i. Population script: populate.js
    c. Upload to database

Best Practice for Simulation Building

- Create bots first
    - be creative but not TOO creative
    - Remember who your target audience and participants will be
    - Be consistent!
    - Use profile images that are not easily on Google Images
    - Don’t be too perfect! Most people don’t fill everything out completely
- Create posts for your bots
    - Get your own images that look “real” and not easily found online
    - Be consistent! Two bots can’t have the same kitchen if you haven’t stated somewhere that they live together, etc

    - Be fun and create a story for your bot. What is their day like? What do they post? What do they like?
    - Spread out timing of posts and follow same rules you listed for participants (remember these bots should look and act like other participants in the study)
    - Create some spam/political infighting/internet drama/web trash as no site is perfect
    - Think about “likes” the posts get, what is popular on the site, what is not
- After posts, make comments on posts
    - be quick and short
    - positive but not too positive
    - role play as your bot, do all your comments for one bot at a time, role-playing as that bot
    - have some small conversations between bots in the comments
    - This is where we create “community”
- Write interactions for bots to participant
    - make it super generic, and comment that could apply to ANY photo
    - make a few, but not too much (no more than what you have created as normal for other posts on the site already)
- Write together
    - If writing simulation collaboratively, upload the CSV files to google docs or box. Makes it much easier to edit together.
- Test test test!
    - Live in your simulation and experience what feels real or fake

Data Collection

Gathering participants and collecting data from your simulation is a critical part in using Truman. While running your study, platforms such as Amazon Mechanical Turk can greatly assist with locating participants, surveying them, and collecting participant data.

What is Amazon Mechanical Turk?

Amazon Mechanical Turk (MTurk) is an online crowdsourcing platform that provides a service for "Requesters," or those who publish tasks on MTurk, to use AI in making requests of humans. These requesters can submit their tasks on the site, approve completed tasks, and ultimately use these completed tasks for their own applications and work. These tasks are completed by "Workers," or individuals who search on the site for tasks and complete them for rewards.

For more information on MTurk and FAQs, visit: https://www.mturk.com/help

Ways you can use MTurk

With MTurk, researchers can more easily gather data by having MTurk workers act as their study participants. Workers can complete requests that researchers submit to the MTurk site as HITs.

You can use MTurk for just about any online task: surveys, collecting data, moderating websites, etc. When running pilot studies, for instance, MTurk can be used to gather feedback from participants as to whether posts on your Truman simulation appear realistic, or to get opinions about how your social media site looks overall.

MTurk can be used to monitor participant usage and activity on your instance of Truman. For instance, the platform can create HITs that collect data from a website. New projects can be created at the following link: https://requester.mturk.com/create/projects/new.

Gathering Participants

Almost anyone can act as a worker on MTurk. The only requirement is to have some kind of computing device and to be over the age of 18. You can determine who exactly you'd like to complete your tasks, based on participant demographics.

As for pricing, the amount Requesters pay is comprised of two parts: the amount paid to the Workers, and the fee paid to MTurk.

For more information on participant pay and pricing, visit:
https://www.mturk.com/pricing

MTurk Best Practices

Finally, when running MTurk studies many common concerns arise such as: gathering eligible participants, receiving quality participant data, ensuring study completion, etc. As such, the following are suggested solutions and best practices for ensuring that you receive quality data in

your Truman study (inspired from Hauser et al.'s "Common Concerns with MTurk as a Participant Tool"):

1) **Ensure attentiveness of participants**
   Make sure your participants stay focused on the task at hand that you are requesting them to do, whether it be them answering your survey carefully or continuously visiting your simulation. The following are suggestions for ensuring this:
   - Restrict your HIT to MTurkers who have a certain HIT approval ratio
   - Include warnings, reminder texts, etc. to motivate participants to pay attention
   - Include comprehension checks, or measure participant response speeds

2) **Make sure participants understand the language that you are administering your study in**
   You should ensure that your participants understand how to complete the HIT. These are some of the things you can do:
   - Restrict eligibility to only those in the geographic regions that are likely to speak/read the language of the simulation, pilot, and surveys fluently
   - Include free response questions to act as comprehension checks

3) **Minimize nonnaivete of participants if you would like manipulations to go undetected**
   If your study contains tasks/manipulations that are sensitive to learning effects of participants(previous exposure to a manipulation, etc.), do the following:
   - Exclude MTurkers who completed your previous HITs
   - Use less popular manipulations/measures
   - Consider limiting the participants of your study to novice/new MTurk workers or other populations

4) **Minimizing deception of participants to prevent participants from inputting misleading or unreliable information**
   If your study relies on sampling a subpopulation of MTurk, and you don't know who belongs in the subpopulation:
   - Pre-screen for the target population
   - Record workerIDs for every survey attempt, so that you can see when a person has submitted multiple responses from the same account

   If your study contains questions with answers that participants can just look up, and you would like to prevent this:
   - Give participant instructions to not look up answers online, and try not to reward them for correct responses
   - Look for evidence of MTurk forums mentioning your HIT, to ensure that participants had not discussed with each other about your study

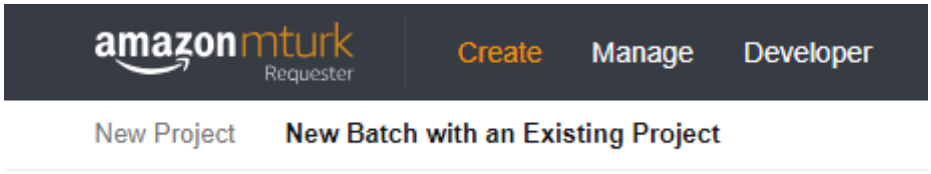
For more details on best MTurk practices, be sure to look at:

David J. Hauser's "Common Concerns with MTurk as a Participant Pool: Evidence and Solutions"

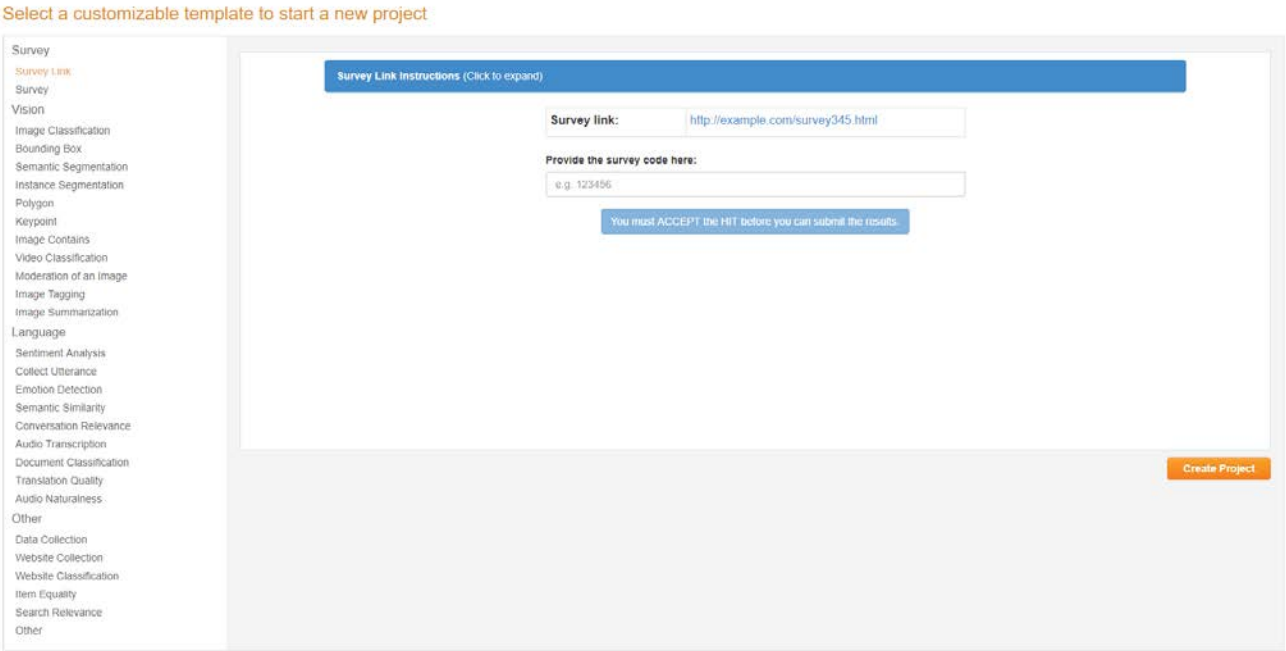
Creating a HIT on Amazon Mechanical Turk (MTurk)

A HIT on MTurk is a "Human Intelligence Task." You are creating a task for the workers to complete. When you build your HIT, you can tell MTurk a few things, including how much you want to pay them, how many people you want to allow to participate in your HIT, and other qualifications you want the workers to meet before they are able to participate.

1. First, you will need a Requester account with Amazon Mechanical Turk
2. If you are creating a new project, click "New Project" in the upper left hand corner



   a.
3. Next you will select what kind of project you will be creating from the list on the left, and then select "Create Project." In this example, we will be using the "Survey link" template.



4. You can edit the instructions and include information about your HIT or things you think you would like the participants to know before beginning.
   a. **Enter Properties:**
      i. Here you will enter the title, description, and key words.
      ii. Setting Up Your Survey:
         1. Enter how much the workers will get paid for the HIT
         2. Enter the number of participants you want (the default is 20). Remember to go back and update this if you launch more than one batch!
         3. The remaining settings deal with how long the HIT will take
      iii. Worker Requirements:

1. Here you can specify whether or not you want the workers to be "Masters." Amazon charges an additional 5% for workers.
2. You can also specify what approval rate you want your workers to have, in addition to number of HITs approved, and worker location.

b. Design Layout:

i. Remember to remove the template note before posting!
ii. Enter the link to your own survey
iii. To make sure participants are actually completing a survey and to help match your MTurk results with your survey results, have Qualtrics generate a random number and ask the workers to enter it before they submit your HIT.

5. Connecting Qualtrics:
    a. Add a random number generator to the survey flow to create a random number for the participants to enter. This random number is recorded in both Qualtrics and MTurk for easier comparison. Follow this link for instructions: http://brentcurdy.net/qualtrics-tutorials/link/
6. Other Tips:
    a. Ask the participants to enter their MTurk ID instead of their name on your consent form
    b. If you are using both MTurk and recruiting students (i.e. SONA) make sure the survey information is correct on both; don't mention class credit on MTurk.

When working with MTurk, it can be important to make it so that people can only participate in your HITs one time. There are a couple of ways to do this--both outlined below.

1. **Generate a javascript to put at the top HIT's HTML**
    a. Link to generate the script: http://uniqueturker.myleott.com/
    b. This can prevent the same worker from taking the same survey again
    c. It can also block participants from future studies-- but you have to make sure you use the ***SAME*** unique identifier across multiple HITs for this function to work
2. **2. Create Qualification Types on MTurk**
    a. This method is a bit more time consuming
    b. Also note that doing this can block participants from future studies but it *cannot* block the same worker from taking the same survey again
    c. For detailed instructions please refer to the appendix in the paper by Sharpe Wessling et al. here.
    d. Note: here we want to do a reverse qualification, so we want to set "has not been granted" value of 1 rather than "equal to"
    e. Some helpful links:
        i. https://www.mturkcrowd.com/threads/how-do-i-give-workers-qualifications-for-taking-my-survey.335/

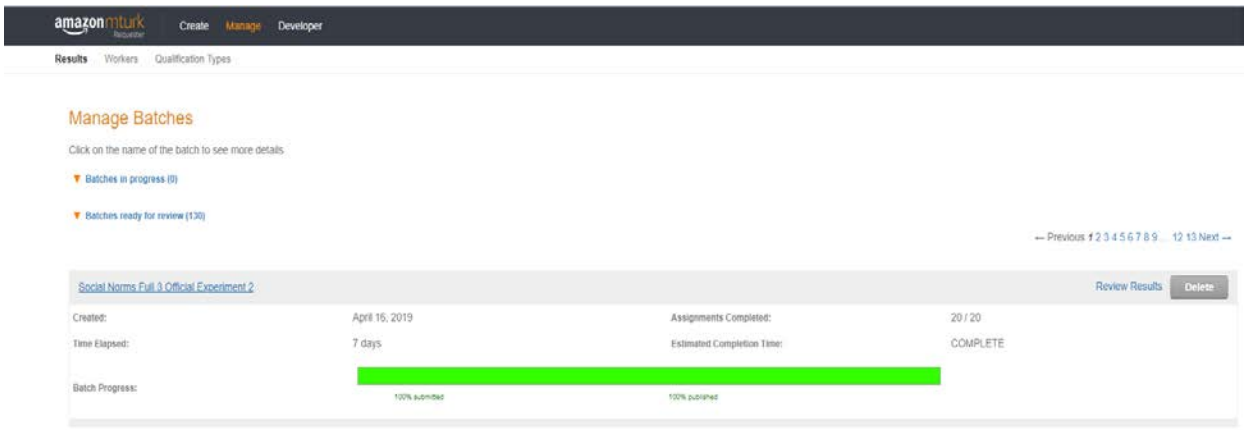
ii. https://blog.mturk.com/tutorial-understanding-requirements-and-qualifications-99a26069fba2

Reviewing the HITs/Payment:

Once the Turkers have completed the task, you will need to review your results. Workers get rewarded once they have completed a task (HIT).

1. In your account, click “Manage” and you will see a list of the various projects and batches you have run (make sure you click on the correct one to review)

2. Then you will click “Review Results” to see a list of the workers who participated



3. If you know you want to approve them all, you can select all and click “approve.” This automatically pays the workers.

**I think including the payment document that Suzanne made here would be helpful

General MTurk Tips/Best Practices:

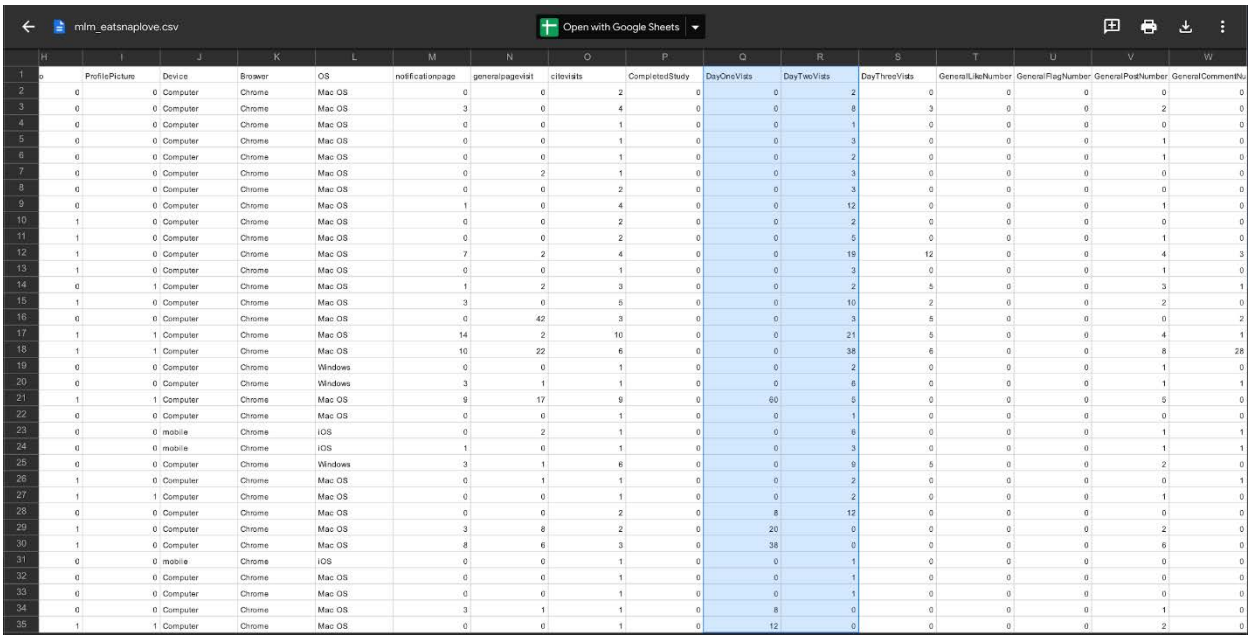
These are some general tips/guidelines for using the Mechanical Turk platform.

1. Set a fair pay rate (in the past NYS minimum wage has been used a guide)
2. Pay promptly!
3. Provide reasonable time estimates.
4. Provide accurate contact information and respond in a timely manner!
5. Approve your results in a consistent and timely manner

PAYMENT

MTurks get rewarded a sum of money when they complete a Human Intelligence Task (HIT).

1. After the study, accept the HIT to automatically reward 50 cents
2. Download csv file from MTurk
3. Open csv file and for each record, check for the following criteria:
    a. DayOneVisits ≥ 2
    b. DayTwoVisits ≥ 2
    c. GeneralPostNumber ≥ 2
4. [Optional] Check other columns (i.e. GeneralLikeNumber, GeneralFlagNumber, etc) for proof of activity
5. If these conditions hold true, pay accordingly:
    a. DayOneVisits => $3
    b. DayTwoVisits => $3
6. If post survey is completed, reward them 50 cents
7. If all parts are completed, reward a bonus of $3, for a total of $10
8. Record amount owed in a new, separate column called “TotalPayment”
9. [Helpful Tip] Highlight as you pay the Turkers and/or have a “Paid” column



	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	o	ProfilePicture	Device	Browser	OS	notificationpage	generalpagevisit	cilevisits	CompletedStudy	DayOneVists	DayTwoVists	DayThreeVists	GeneralLikeNumber	GeneralFlagNumber	GeneralPostNumber	GeneralCommentNu
2	0	0	Computer	Chrome	Mac OS	0	0	2	0	0	2	0	0	0	0	0
3	0	0	Computer	Chrome	Mac OS	3	0	4	0	0	8	3	0	0	2	0
4	0	0	Computer	Chrome	Mac OS	0	0	1	0	0	1	0	0	0	0	0
5	0	0	Computer	Chrome	Mac OS	0	0	1	0	0	3	0	0	0	1	0
6	0	0	Computer	Chrome	Mac OS	0	0	1	0	0	2	0	0	0	1	0
7	0	0	Computer	Chrome	Mac OS	0	2	1	0	0	3	0	0	0	0	0
8	0	0	Computer	Chrome	Mac OS	0	0	2	0	0	3	0	0	0	0	0
9	0	0	Computer	Chrome	Mac OS	1	0	4	0	0	12	0	0	0	1	0
10	1	0	Computer	Chrome	Mac OS	0	0	2	0	0	2	0	0	0	0	0
11	1	0	Computer	Chrome	Mac OS	0	0	2	0	0	5	0	0	0	1	0
12	1	0	Computer	Chrome	Mac OS	7	2	4	0	0	19	12	0	0	4	3
13	1	0	Computer	Chrome	Mac OS	0	0	1	0	0	3	0	0	0	1	0
14	0	1	Computer	Chrome	Mac OS	1	2	3	0	0	2	5	0	0	3	1
15	1	0	Computer	Chrome	Mac OS	3	0	5	0	0	10	2	0	0	2	0
16	0	0	Computer	Chrome	Mac OS	0	42	3	0	0	3	5	0	0	0	2
17	1	1	Computer	Chrome	Mac OS	14	2	10	0	0	21	5	0	0	4	1
18	1	1	Computer	Chrome	Mac OS	10	22	6	0	0	38	6	0	0	8	28
19	0	0	Computer	Chrome	Windows	0	0	1	0	0	2	0	0	0	1	0
20	0	0	Computer	Chrome	Windows	3	1	1	0	0	6	0	0	0	1	1
21	1	1	Computer	Chrome	Mac OS	9	17	9	0	60	5	0	0	0	5	0
22	0	0	Computer	Chrome	Mac OS	0	0	1	0	0	1	0	0	0	0	0
23	0	0	mobile	Chrome	iOS	0	2	1	0	0	6	0	0	0	1	1
24	0	0	mobile	Chrome	iOS	1	0	1	0	0	3	0	0	0	1	1
25	0	0	Computer	Chrome	Windows	3	1	6	0	0	9	5	0	0	2	0
26	1	0	Computer	Chrome	Mac OS	0	1	1	0	0	2	0	0	0	0	1
27	1	1	Computer	Chrome	Mac OS	0	0	1	0	0	2	0	0	0	1	0
28	0	0	Computer	Chrome	Mac OS	0	0	2	0	8	12	0	0	0	0	0
29	1	0	Computer	Chrome	Mac OS	3	8	2	0	20	0	0	0	0	2	0
30	1	0	Computer	Chrome	Mac OS	8	6	3	0	38	0	0	0	0	6	0
31	0	0	mobile	Chrome	iOS	0	0	1	0	0	1	0	0	0	0	0
32	0	0	Computer	Chrome	Mac OS	0	0	1	0	0	1	0	0	0	0	0
33	0	0	Computer	Chrome	Mac OS	0	0	1	0	0	1	0	0	0	0	0
34	0	0	Computer	Chrome	Mac OS	3	1	1	0	8	0	0	0	0	1	0
35	1	1	Computer	Chrome	Mac OS	0	0	1	0	12	0	0	0	0	2	0

	L	M	N	O	P	Q	R	S	T	U	V	
1	OS	notificationpage	generalpagevisit	citevisits	CompletedStudy	DayOneVists	DayTwoVists	DayThreeVists	GeneralLikeNumber	GeneralFlagNumber	GeneralPostNumber	Genera
2	Mac OS	0	0	2	0	0	2	0	0	0	0	
3	Mac OS	3	0	4	0	0	8	3	0	0	2	
4	Mac OS	0	0	1	0	0	1	0	0	0	0	
5	Mac OS	0	0	1	0	0	3	0	0	0	1	
6	Mac OS	0	0	1	0	0	2	0	0	0	1	
7	Mac OS	0	2	1	0	0	3	0	0	0	0	
8	Mac OS	0	0	2	0	0	3	0	0	0	0	
9	Mac OS	1	0	4	0	0	12	0	0	0	1	
10	Mac OS	0	0	2	0	0	2	0	0	0	0	
11	Mac OS	0	0	2	0	0	5	0	0	0	1	
12	Mac OS	7	2	4	0	0	19	12	0	0	4	
13	Mac OS	0	0	1	0	0	3	0	0	0	1	
14	Mac OS	1	2	3	0	0	2	5	0	0	3	
15	Mac OS	3	0	5	0	0	10	2	0	0	2	
16	Mac OS	0	42	3	0	0	3	5	0	0	0	
17	Mac OS	14	2	10	0	0	21	5	0	0	4	
18	Mac OS	10	22	6	0	0	38	6	0	0	8	
19	Windows	0	0	1	0	0	2	0	0	0	1	
20	Windows	3	1	1	0	0	6	0	0	0	1	
21	Mac OS	9	17	9	0	60	5	0	0	0	5	
22	Mac OS	0	0	1	0	0	1	0	0	0	0	
23	iOS	0	2	1	0	0	6	0	0	0	1	
24	iOS	1	0	1	0	0	3	0	0	0	1	
25	Windows	3	1	6	0	0	9	5	0	0	2	
26	Mac OS	0	1	1	0	0	2	0	0	0	0	
27	Mac OS	0	0	1	0	0	2	0	0	0	1	
28	Mac OS	0	0	2	0	8	12	0	0	0	0	
29	Mac OS	3	8	2	0	20	0	0	0	0	2	
30	Mac OS	8	6	3	0	38	0	0	0	0	6	
31	iOS	0	0	1	0	0	1	0	0	0	0	
32	Mac OS	0	0	1	0	0	1	0	0	0	0	
33	Mac OS	0	0	1	0	0	1	0	0	0	0	
34	Mac OS	3	1	1	0	8	0	0	0	0	1	
35	Mac OS	0	0	1	0	12	0	0	0	0	2	

100% $ % .0 .00 123 Arial 10

fx notificationpage

	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	notificationpage	generalpagevisit	citevisits	CompletedStudy	DayOneVists	DayTwoVists	DayThreeVists	GeneralLikeNum	GeneralFlagNum	GeneralPostNum	GeneralCommentNumber		TotalPayment	Paid
2	0	0	2	0	0	2	0	0	0	2	0		$4	Yes
3	3	0	4	0	3	8	3	0	0	2	0		$6.50	No
4	0	0	1	0	4	1	0	0	0	4	0		$10.00	No
5	0	0	1	0	0	3	0	0	0	1	0			
6	0	0	1	0	0	2	0	0	0	1	0			
7	0	2	1	0	0	3	0	0	0	2	0			
8	0	0	2	0	0	3	0	0	0	0	0			
9	1	0	4	0	0	12	0	0	0	1	0			
10	0	0	2	0	0	2	0	0	0	0	0			
11	0	0	2	0	0	5	0	0	0	1	0			
12	7	2	4	0	0	19	12	0	0	4	3			
13	0	0	1	0	0	3	0	0	0	1	0			
14	1	2	3	0	0	2	5	0	0	3	1			
15	3	0	5	0	0	10	2	0	0	2	0			
16	0	42	3	0	0	3	5	0	0	0	2			
17	14	2	10	0	0	21	5	0	0	4	1			
18	10	22	6	0	0	38	6	0	0	8	28			
19	0	0	1	0	0	2	0	0	0	1	0			
20	3	1	1	0	0	6	0	0	0	1	1			
21	9	17	9	0	60	5	0	0	0	5	0			
22	0	0	1	0	0	1	0	0	0	0	0			
23	0	2	1	0	0	6	0	0	0	1	1			
24	1	0	1	0	0	3	0	0	0	1	1			
25	3	1	6	0	0	9	5	0	0	2	0			
26	0	1	1	0	0	2	0	0	0	0	1			
27	0	0	1	0	0	2	0	0	0	1	0			
28	0	0	2	0	8	12	0	0	0	0	0			
29	3	8	2	0	20	0	0	0	0	2	0			
30	8	6	3	0	38	0	0	0	0	6	0			
31	0	0	1	0	0	1	0	0	0	0	0			
32	0	0	1	0	0	1	0	0	0	0	0			