MTurk Framework Requirements Document

By **Akshat, Jay, Jonathan, Muhammad, Walker, Xinjian**

February 6, 2021

Executive Summary

The Mechanical Turk Experiment Framework is a software development project that involves building a web application that integrates with the Amazon Mechanical Turk API. The goal of this project is to provide our client with a simple and effective framework for running and managing web-based experiments (otherwise known as Human Intelligence Tests) on Amazon Mechanical Turk. Currently, the system used by our client is lacking many of the key features that they require, such as the ability to easily pay bonuses or an effective functional dashboard for managing their experiments. Existing software that does contain some of this required functionality either doesn't integrate well with MTurk or is built upon a deprecated API, so they need an alternative solution to meet their needs. Our proposed system will provide them with a tool that contains the core functionalities that they require, while also providing them with a solid infrastructure that is secure and scalable into the future. Users of this system fall into two categories: administrators (or requesters in terms of MTurk) and turkers (or workers in terms of MTurk). Administrators consist of researchers within the computing science department at the University of Alberta. These users will be responsible for managing the application and supplying HITs. Turkers consist of the Amazon Mechanical Turk workers who will be partaking in said HITs.

Project Glossary

- **Administrator**: a requester customer that creates the HITs in order for it to be completed by Turkers.
- **Assignment**: each Turker has an assignment for a given HIT.
- **Batch**: a set of HITs.
- **Experiment**: the overlying logical structure that the administrator (researcher) is interested in. In our case, an experiment can be made up of many HITTypes.
- **Game**: another logical construct that the administrator is interested in. A treatment is made up of multiple games, and these games contain the actual experimental data.
- **HIT**: a single, self-contained, virtual task, also known as a Human Intelligence Task.

- **HITTypes**: a set of common properties for HITs in the Mechanical Turk Framework. Also referred to as a template for HITs.
- **Lobby**: an environment where a single person or a group of people can perform the HIT. Also referred to as a treatment.
- **MTurk**: a marketplace for completion of virtual tasks (HITs) that requires human intelligence, also known as Mechanical Turk. Consists of requesters that creates the tasks and Turkers that complete these tasks.
- **Requesters**: people who create and provide Human Intelligence Tasks through Amazon Mechanical Turk.
- **Treatment**: another logical construct that the administrator is interested in. An experiment has multiple treatments, and these treatments can be divided into multiple HITs.
- **Turker**: a worker customer that works on a HIT, submit an answer, and collects a reward for completion.
- **User**: anyone using the system, both parties (administrators and turkers) being included.
- **Worker**: synonymous with the term turker.

User stories

We want to note that we realize that some of these user stories are handled by Amazon Mechanical Turk system and API and not specifically by our application. However, we would like to keep these user stories to help serve as a guide to our end goal (*SH1*).

Must have:

MH1.
As an administrator, I want to be able to login to my account, so that I can securely access the website on my account.

- *Given*: admin attempts to log in with their user details
- *When*: admin is prompted to login on the login dashboard (username, password)
- *Then*: login is authorized and admin is logged in

MH2.

As an administrator, I want to be able to view assignments on the dashboard, so that I can easily see the status of assignments.

- *Given*: a list of assignments exists in the system
- *When*: admin goes to assignment dashboard
- *Then*: admin is able to view assignments and the status of them

MH3.

As an administrator, I want to be able to create HITTypes, so that HITs can be grouped by HITTypes.

- *Given*: an idea/template for a HITType to be added to the system
- *When*: admin is on the HITTypes section of the dashboard
- *Then*: admin can create HITTypes accordingly from the template

MH4.

As an administrator, I want to be able to assign HITs to HITTypes, so that I have control over the data for my experiments.

- *Given*: a set of HITs that the admin wants to assign a HITType to
- *When*: admin is on the HITs section of the dashboard
- *Then*: admin can manage/assign HITs to HITTypes accordingly for data

MH5.

As an administrator, I want to be able to generate HITs from HITTypes, so that I can set up assignments.

- *Given*: a concept/idea for a HIT the admin wants to create
- *When*: admin is on the HITs section of the dashboard
- *Then*: admin can generate HITs

MH6.

As an administrator, I want to be able to manage HITs, so that I can modify my assignments.

- *Given*: HITs exist in the system that the admin wants to modify
- *When*: admin is on the HITs sections of the dashboard
- *Then*: admin can manage/modify HITs accordingly

MH7.

As an administrator, I want to be able to review HITs, so that I can analyze my assignments.

- *Given*: HITs exist in the system that the admin wants to review
- *When*: admin is on the HITs section of the dashboard
- *Then*: admin can analyze/review HITs accordingly

MH8.

As an administrator, I want to be able to assign qualifications for Turkers in bulk, so that I can manage the restrictions for Turkers with ease.

- *Given*: Turker(s) exist in the system and the admin wants to assign them qualifications
- *When*: admin is in the Dashboard
- *Then*: admin can allow Turker(s) to do assignments based on qualifications

MH9*.*

As an administrator, I want to be able to remove qualifications for Turkers in bulk, so that I can manage the restrictions for Turkers with ease.

- *Given*: Turker(s) exist in the system and the admin wants to remove their qualifications
- *When*: admin is in the Dashboard
- *Then*: admin can restrict Turker(s) ability to do assignments based on qualifications

MH10*.*

As an administrator, I want to be able to have a bulk bonus payment option, so that I can pay bonuses to multiple Turkers with ease.

- *Given*: a list of completed assignments in the system
- *When*: admin views the batch in the dashboard
- *Then*: admin can choose to pay bonuses to all eligible Turkers who completed assignments in the batch in bulk.

Should have:

SH2*.*

As an administrator, I want to be able to create a lobby, so that multiple Turkers can complete a multi-user HIT simultaneously.

- *Given*: a concept/idea for a HIT that requires multiple workers
- *When*: admin is creating HITs/lobbies in the dashboard
- *Then*: Turkers can complete a multi-user HIT in given lobby simultaneously

SH4*.*

As an administrator, I want to be able to use the search bar, so that I can search for any assignments.

- *Given*: a list of assignments exists in the system
- *When*: admin clicks on search bar and enters search terms
- *Then*: system displays assignments based on desired search results

SH5*.*

As an administrator, I want to be able to filter the search bar, so that I can filter out specific assignments.

- *Given*: a list of assignments exists in the system
- *When*: admin clicks on filter button to filter by desired category
- *Then*: system displays assignments based on admin ’s filter

Could have:

CH1*.*

As an administrator, I want to be able to see system status, so that I know what to do when the system gives any error.

- *Given*: admin is logged into the system
- *When*: admin encounters error
- *Then*: admin sees system status for given error

CH2*.*

As a user, I want to be able to see help documentation for features and functions of the web app, so that I know how to use the website

- *Given*: user has a question about the overall features/functions of web app
- *When*: sser clicks on help documentation
- *Then*: user gets information on how to use the website and how certain features work

Would like but won't get:

WL1*.*

As an administrator, I want to be able to automatically approve and pay bonuses based on certain conditions, so that I can pay bonuses to Turkers with ease.

- *Given*: a list of completed assignments exists in the system
- *When*: HITs are completed based on certain conditions (i.e. Turker qualifies for a bonus)
- *Then*: the system will approve and pay bonuses to them automatically

Similar products

- TurkServer (https://github.com/TurkServer/long-run-cooperation)
    - This is an open-source Turkserver demo that contains much of the functionality that we are going to need. It does not handle paying Turkers or managing assignments. It does show how experiments can have different treatments, and some experiments contain multiple tasks to be done by the Turker. We can draw inspiration from this product and use it as a source of information to implement many of Must Have and Should Have functionalities from our user stories.
- psiTurk (http://psiturk.org/)
    - From their website: "psiTurk takes the hassle out of collecting behavioral data online. Test your code, post HITs, serve secure ads, pay participants. psiTurk lets you focus on your research rather than software development." This is an

open tool that also contains much of the functionality that we are looking to implement in our system. Unlike the long-run-cooperation tool, this tool can handle paying Turkers, and will likely help when we start working on payment logic.

Open-source products

- Turkserver (https://turkserver.readthedocs.io/en/latest/index.html)
    - TurkServer is a platform for building software-controlled, web-based behavioral experiments using participants from the Internet. There are a ton of similar tools built on this platform that we can reference while working on the project.
- oTree (https://www.otree.org/)
    - oTree lets you create controlled behavioral experiments, multiplayer strategy games, surveys and quizzes. This product will help us better understand the games that are mentioned throughout the project.
- Pybossa (https://pybossa.com/)
    - An open source alternative to Amazon Mechanical Turk.
- https://github.com/mitcho/turktools
    - Tools for preparing linguistic surveys for Amazon Mechanical Turk (AMT). This tool looks to be deprecated, but it is still useful for analysis.

Technical resources

- Amazon Mechanical Turk API (https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/Welcome.html) (https://docs.aws.amazon.com/cli/latest/reference/mturk/index.html)
    - Amazon Mechanical Turk is the system we will be integrating with using the Amazon Mechanical Turk API.
- CSS
    - A frontend language that will be used to improve the look and flow of our UI.
- Django (https://www.djangoproject.com/)
    - This is the web framework that we will be using to build our backend using python.
- Django Rest Framework (https://www.django-rest-framework.org/)
    - Django REST framework is a powerful and flexible toolkit for building Web APIs.

- HTML (https://html.com/)
    - A simple frontend language that every web application uses.
- Javascript (https://www.javascript.com/)
    - A language used to improve the interactivity and usability of our web application.
- Material-ui (https://material-ui.com/)
    - A frontend component for React to efficiently design our system.
- Material-table (https://material-table.com/#/)
    - React data table component to design our table dashboards with various optional functionalities.
- pgAdmin (https://www.pgadmin.org/)
    - Open source administration and development platform for PostgreSQL; handles our database
- PostgreSQL (https://www.postgresql.org)
    - The preferred database management system of our client.
- Python (https://www.python.org/)
    - The main programming language that will be used in our backend. It's easy to understand, maintain and extend for future developments.
- React (https://reactjs.org/)
    - A JavaScript library for building user interfaces in our frontend.
- React Bootstrap (https://react-bootstrap.github.io/)
    - A frontend framework for React to efficiently design our system.
- Webpack (https://webpack.js.org/)
    - Used to compile and bundle our modules for usage in a single browser (i.e., babel-loader); used to make our django-react integration happen