# 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 and existing software that does contain these features is deprecated, 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 framework infrastructure that is extendable and scalable in case of future expansion. Users of this system fall into two categories, and the key functional requirements of this project are also classified based on these two types of users. The first type of users are “administrators”, who consist of researchers within the computing science department at the University of Alberta. These users will be responsible for managing the application and providing HITs. The other type of users are “Turkers”, and these users consist of the Amazon Mechanical Turkers 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. Assignment and HIT may be used synonymously throughout the project.
- **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 administrators that creates the tasks and Turkers that complete these tasks.
- **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 collect a reward for completion.
- **User**: anyone using the system, both parties (administrators and turkers) being included.

# User stories

## Must have:

1. As a user, I want to be able to login to my account, so that I can securely access the website on my account.
    - Given: login details (username, password); login dashboard
    - When: User is prompted to login using login details (username, password)
    - Then: login is authorized and User is logged in
2. As a user, I want to be able to view assignments on the dashboard, so that I can easily preview what assignments are there.
    - Given: the assignments dashboard
    - When: User goes to assignment dashboard
    - Then: User is able to view their assignments
3. As an administrator, I want to be able to approve assignments on the dashboard, so that I can restrict what assignments can go through.
    - Given: the assignments dashboard
    - When: Admin reviews the assignments
    - Then: puts restrictions on what assignments can be submitted

4. As an administrator, I want to be able to create HITTypes, so that HITs can be grouped by HITTypes.
    - Given: a template
    - When: Admin are on the HITType dashboard
    - Then: Admins can create HITTypes accordingly from the template
5. As an administrator, I want to be able to assign HITTypes, so that I have control over the data for my experiments.
    - Given: HITTypes
    - When: Admin are on the HITType dashboard
    - Then: Admin can manage/assign HITTypes accordingly for data
6. As an administrator, I want to be able to create HITs, so that I can set up my assignments.
    - Given: an experimental concept
    - When: Admins are on the HITs dashboard
    - Then: Admin can create HITs
7. As an administrator, I want to be able to manage HITs, so that I can modify my assignments.
    - Given: HITs
    - When: Admins are on the HITs dashboard
    - Then: Admins can manage/modify HITs accordingly
8. As an administrator, I want to be able to review HITs, so that I can analyze my assignments.
    - Given: HITs
    - When: Admins are on the HITs dashboard
    - Then: Admins can analyze/review HITs accordingly
9. 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: a Turker and their qualifications
    - When: lobbies are created
    - Then: Admins can restrict lobbies to Turkers who have the qualifications
10. 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: a Turker and their qualifications
    - When: lobbies are created

- Then: Admins can modify/remove qualifications in the given lobby

11. As a turker, I want to be able to see what qualifications are needed to do an assignment for a given HIT, so that I know if I am eligible to complete it.
    - Given: qualifications for an assignment for a given HIT
    - When: Turker views assignment and sees qualifications
    - Then: Turker knows if eligible to do that assignment based on qualifications
12. As an administrator, I want to be able to have a bulk payment option, so that I can pay multiple Turkers with ease.
    - Given: a completed batch
    - When: Admin goes to approve the batch
    - Then: Admins can pay all the Turkers who completed the HITs in the batch in bulk.

## Should have:

1. As a turker, I want to be able to accept my payment without any unnecessary steps, so that I can receive my payment with ease.
    - Given: completed approved tasks from Admin
    - When: on Turker's dashboard
    - Then: Turker can receive payment
2. As an administrator, I want to be able to create a lobby, so that multiple turkers can complete a HIT in a lobby simultaneously.
    - Given: Multiple Turkers
    - When: Admin clicks to create lobby
    - Then: Turkers can complete HIT in given lobby at same time
3. As a turker, I want to be able to join a lobby, so that I can complete the multiplayer HITs.
    - Given: Lobby
    - When: Turker clicks to join lobby
    - Then: Turker can complete the multiplayer HITs in the given lobby
4. As a user, I want to be able to use the search bar, so that I can search for any assignments.
    - Given: web app
    - When: User clicks on search bar and enters search terms
    - Then: system outputs desired search results

5. As a user, I want to be able to filter the search bar, so that I can filter out specific assignments.
    - Given: web app
    - When: User clicks on filter button by desired category
    - Then: system outputs desired search results based on User's filter

## Could have:

1. As a Turker, I want to be able to play a turn-based game simultaneously, so that it's efficient for both parties.
    - Given: HITs
    - When: Turker is completing the HIT
    - Then: both parties will play the game at the same time
2. As a user, I want to be able to see system status, so that I know what to do when the system gives any error.
    - Given: web app
    - When: User encounters error
    - Then: User sees system status for given error
3. 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: overall features/functions of web app
    - When: User clicks on help documentation
    - Then: User gets information on how to use the website accordingly

## Would like but won't get:

1. 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: Turkers with completed HITs
    - When: HITs are completed based on certain conditions
    - Then: system will approve and pay them automatically
2. As a user, I want to be able to send messages, so that both parties have direct contact with ease.
    - Given: 2 users
    - When: Users needs to communicate and be in contact
    - Then: Users can send messages to each other directly

## 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.
- pgAdmin (https://www.pgadmin.org/)
    - Open source administration and development platform for PostgreSQL; handles our database
- Django (https://www.djangoproject.com/)
    - This is the web framework that we will be using to build our backend using python.
- 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.
- 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.
- Material-ui (https://material-ui.com/)
    - A frontend component for React to efficiently design our system.