Modifiable Multi-Tasking Environment (ModME)

User Manual

27 September 2022

Prepared by
Beth M. Hartzler, PhD

Prepared for
Air Force Research Laboratory
2620 Q Street, Bldg 852
Wright-Patterson AFB, OH 45433

MODIFIABLE MULTI-TASKING ENVIRONMENT (MODME) USER MANUAL

Prepared by

Beth M. Hartzler, PhD
CAE USA

Ryan Remaly
CAE USA

James Cline
CAE USA

Robert Theimer
Cubic

Patrick Dull
CAE USA

The contractor team would also like to acknowledge the contributions of Leslie M. Blaha, PhD, Air Force Research Laboratory and Leif Carlsen, Pacific Northwest National Laboratory, for their work in preparing the initial ModME program and user guides.

Section **TABLE OF CONTENTS** **Page**

Figure **LIST OF FIGURES** **Page**

1.0 GENERAL DOCUMENTATION AND USER MANUAL

1.1 Introduction

The goal of this project is to create a fully-customizable, multi-tasking experimental environment. The Modifiable Multitasking Environment (ModME) provides a simple-to-use interface for customizing and conducting research involving vigilance, workload, and strategic human behavior using operationally relevant tasking. In ModME, "condition" refers to a set of tasks that participants complete and for which data are collected. Conditions are distinguished by the variety of tasks included, duration, task specifications, and other traits. The word "experiment" in this context refers to a data collection session involving one or more conditions, though these experimental sessions may be incorporated into a larger study involving other measures. Collected data are organized according to a series of experimental parameters.

ModME is built in JavaScript, Hypertext Markup Language (HTML), and Cascading Style Sheets (CSS) on top of a Structured Query Language Lite (SQLite) database, allowing for simple storage of experimental conditions and data. Following the steps outlined below will enable researchers to download the necessary software and resources, and establish a local ModME host site (i.e., webpage that serves as the base for the server). The host site will be *http://<ip>:<port*>, where <*ip*> will be either the Internet Protocol (IP) address of the server or a domain name that points to the server. The other part, <port>, will be the port specified when launching the server, default 9000.

1.1.1 Reference Materials

- Django https://www.djangoproject.com/download/
- Original Multi-Attribute Test Battery (MATB) http://matb.larc.nasa.gov/

1.1.2 Required Software

This list is composed of software and libraries needed for ModME. The version numbers here are the ones used in the latest release of ModME.

- Python 3.x, via Anaconda Distribution – https://www.anaconda.com/products/distribution#all
- Django 2.2.7 – https://www.djangoproject.com/
- South 0.8.3 – https://south.readthedocs.org/en/latest/
- Tornado 6.0.3 – https://www.tornadoweb.org/en/stable/
- ModME download – https://github.com/SmartAdaptiveInterfaces/modme3
- Mac Ports – https://www.macports.org/install.php [*needed only when installing on Macintosh operating system* (macOS)]

This software has primarily been used and tested with Google Chrome, but any up-to-date browser should work.

1.2 Installation and Setup

Installation instructions can be found in the README.md file in the ModME source folder.

1.2.1 Setup ModME Environment

Place the ModME folder in an easy to reference location on the machine or server. The Desktop is used in the example, but this location may be within the virtual environment (VE) setup as suggested in the README documentation, or another desired location.

Check prerequisites in the terminal or command line (note that >$ will already be on each command line and does not need to be retyped):

```
>$ python3 –version >$ pip3 –version >$ tar –version >$ virtualen
v –version
```

Create a VE to contain the server (optional but recommended); these are command line commands for creating a VE called "virtmodme" which preserves the specific versions of libraries used within that VE. Using this will streamline launching the ModME server in the future because the needed library versions are set up in the VE.

`$ virtualenv virtmodme $ cd virtualmodme/` [*this moves into the virtual environment*]

`$ source bin/activate` [*Mac/Linux*]

`$ Scripts` [*Windows*]

Install Python library dependencies inside the VE:

```
$ pip3 install Django==2.2.12 $ pip3 install tornado==9.0.3 d
```

Install and configure the ModME server locally by unzipping the modme.zip folder inside the VE.

1.2.2 Start Server

If not already in the terminal, change the present working directory to the ModME folder inside the VE created above by inputting:

```
>$ cd Desktop/virtmodme/modme
```

Launch the server by running the following command in the terminal:

```
>$ python3 Tornado.py
```

By default, the server will run on port 9000.

The port can be changed by adding a parameter to the command with the appropriate port (e.g., "python Tornado.py 8888", which will launch the server on port 8888).

If everything worked correctly, text appearing in the command prompt should be similar to that displayed in sample terminal shown in Figure 1, though the exact library version numbers may vary by setup.

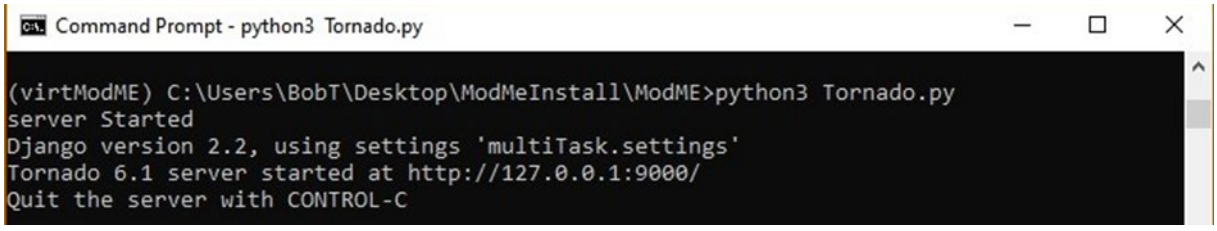


Figure 1. Message in Terminal for Correctly Started Server

1.2.3 Accessing the Experiment

ModME can now be accessed using a client computer at host-site/MATB, the landing page as shown in Figure 2.

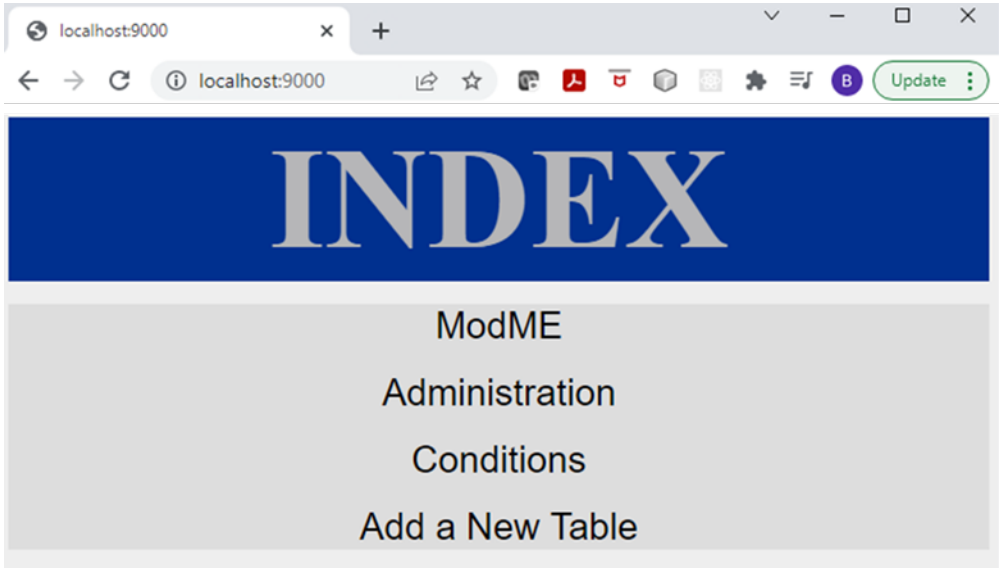


Figure 2. Main Landing Page for ModME

1.3 System Overview

ModME is composed of a server and client setup. The client setup is where the actual experiment will run and can be used on any computer connected to the server via a network. The server hosts all code and stores the database containing the experimental conditions and data. The server is written in Python using the Django and South packages. Once running, the server hosts a series of webpages that can be accessed by a client computer. The two major webpages for the site are the Admin and Experiment page.

The basic ModME task environment contains one to four unique tasks designed to test cognitive workload and situational awareness (Figure 3). The tasks are self-contained in the four quadrants of the screen and are described in the following four subsections. Note that ModME allows each task to be flexibly implemented, giving researchers extensive control over the configuration of each task and the experimental workspace as a whole. From left to right, top to bottom, the tasks shown in Figure 3 are Monitoring, Tracking, Visual Communications, and Resource Management.

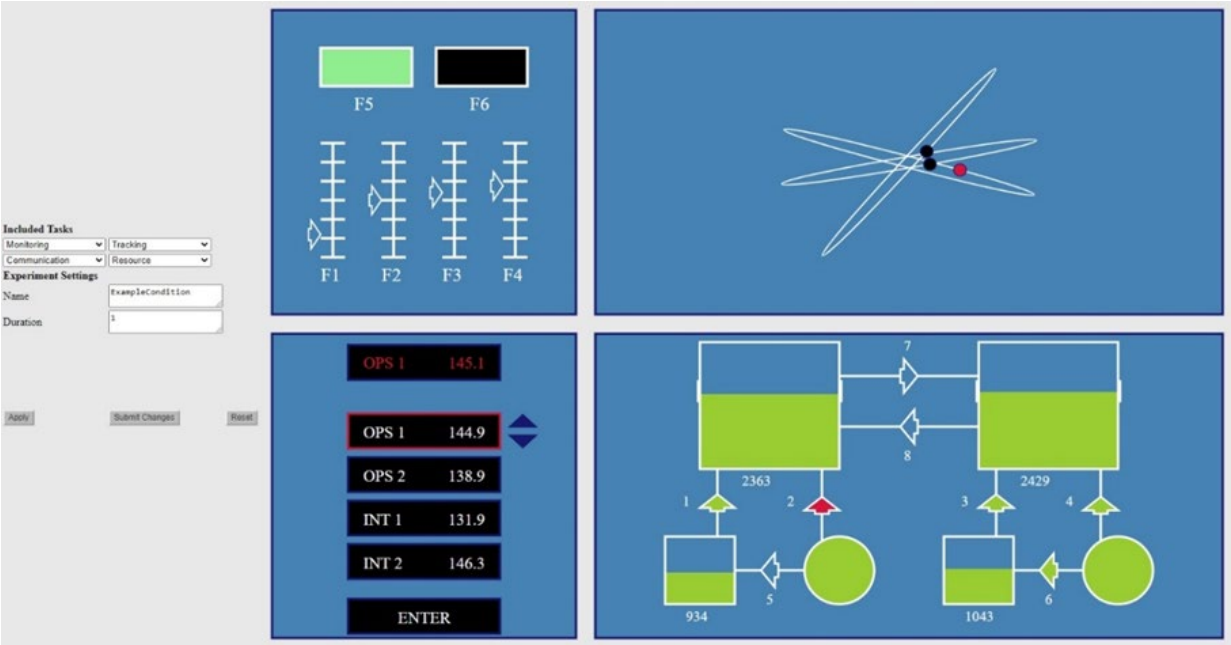


Figure 3. Four Sample Tasks in ModME Experiment Environment

1.3.1 Instructions

As part of the customizable testing environment, this version of ModME does not include any standardized instructions to participants. If researchers want to present either the informed consent document (ICD) or task instructions electronically, rather than as a hard copy, the desired information can be customized and added when creating or modifying the experimental controls. This information for participants can either be presented as standard HTML text on the screen or included as a PDF.

1.3.2 Tracking Test

The Tracking Task is usually located in the upper-right quadrant of the ModME environment, as illustrated in Figure 4. On the screen, a set of colored dots representing satellites are continually moving along pre-defined trajectories, depicted by white lines. The default task settings consist of three oval trajectories. Randomly, one of the satellites will turn red, indicating an alert. The correct response is for the participant to click on the red dot, which then turns green, meaning "target acquired". The participant should then continuously track the green dot with the mouse. Tracking means keeping the mouse cursor (arrow) on top of the target while it traces its path; the participant does not need to continuously hold the mouse button while tracking the target. Use of

a keyboard to enter responses to other tasks presented in conjunction with this Tracking task will enable participants to continue tracking the satellites with a mouse. When any target dot turns red, the participant acquires and tracks this target. Further detail on configuration of this task may be found in Appendix A or the Condition Administration section (5.0).

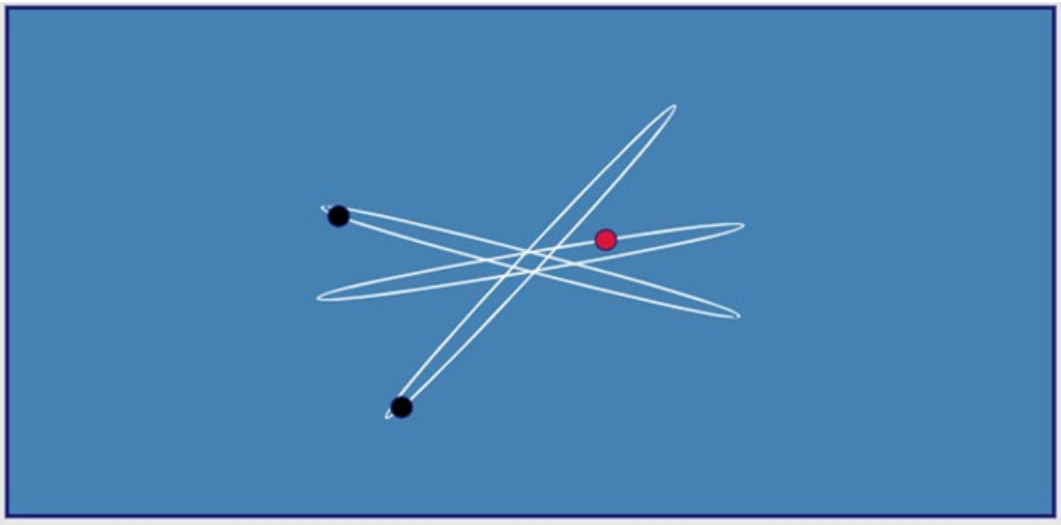


Figure 4. Tracking Test with Three Satellites

1.3.3 Crosshair Tracking (Object Control) Task

The Crosshair Tracking task is an alternative tracking task, emulating the tracking/object control task in the MATB as close as possible with browser constraints.

In this task, the quadrant contains a two-axis crosshair of yellow plus (+) signs. A green cursor jitters around the quadrant. The cursor starts at the center of the crosshair and moves around with a configurable jitter speed. A yellow region is defined by either a circle (as shown in Figure 5) or square. The goal is to keep the cursor inside the yellow region without crossing the border. The participant can move the cursor back to the center of the screen by clicking and holding on the cursor, dragging to the desired position and releasing the mouse or joystick button (Figure 5). Further detail on configuration of this task may be found in Appendix A or the Condition Administration section (5.0).

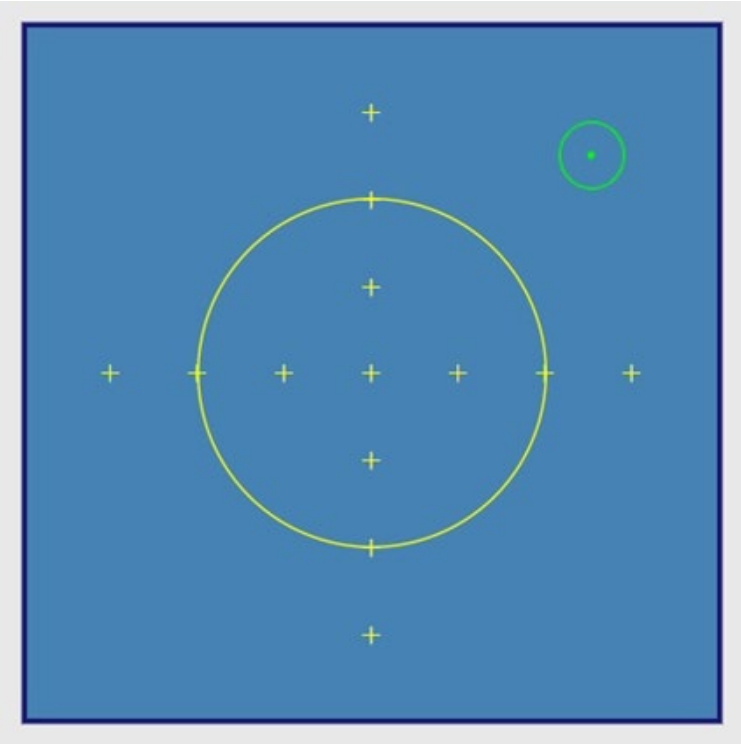


Figure 5. Crosshair Tracking Task with Cursor Outside Target Boundary

1.3.4 Monitoring Task

The goal of the Monitoring Task is to track the gauges to recognize and respond to any alerts. Alerts are indicated when either a button (label 1 in Figure 6) changes color or one of the sliders (label 2) moves beyond the target range. The participant's goal is to reset the object (i.e., button or slider icon) within a certain amount of time, after which the object will return to its base state.

The Monitoring Task attributes that can be modified include the number of sliders, speed of the sliders, number of colored blocks and the frequency of color alerts. Further details on these options may be found in Appendix A or the Condition Administration section (5.0).

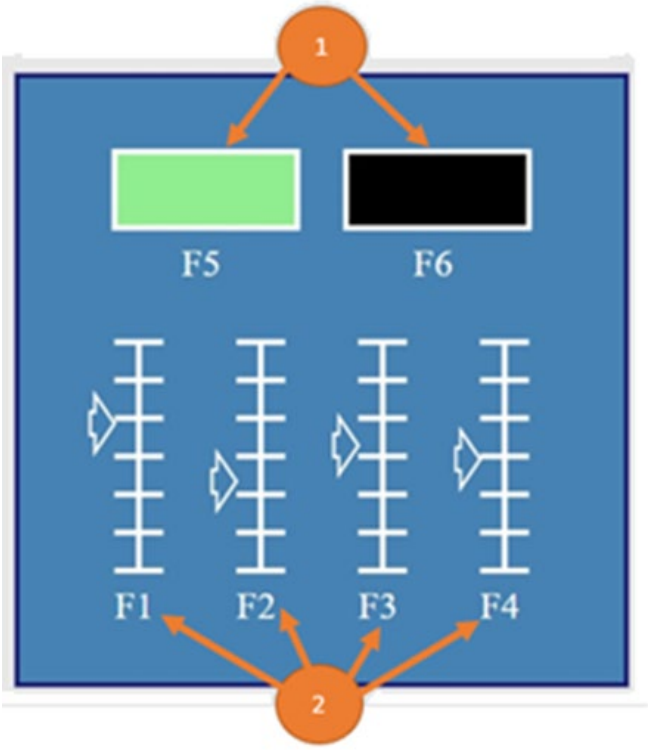


Figure 6. Monitoring Task

1.3.5 Colored Blocks

At the top of the quadrant, the Monitoring Task has two colored blocks which indicate when an alert is present by changing colors. The green block on the left can randomly turn black and must be reset by pressing F5. The black block on the right can randomly turn red and must be reset by pressing F6. Data collected on the colored blocks includes the timestamp and input buttons of the keyboard presses. If a color is not reset with a button press, it will eventually time out and return to its normal color state and a miss is recorded in the data.

1.3.6 Vertical Sliders

A set of vertical slides are shown on the screen, consisting of a vertical bar with hatch marks and moving triangle sliders. The normal range of motion of the sliders is +/- 1 from the center hatch mark. A slider becomes alert if it moves outside the normal range of motion. If a slider moves

above or below this normal range around the center, the participant must push the F1-F4 buttons associated with that slider to reset the position of the triangle. Upon a key press, the triangle returns to the 0 mark, or the center of the vertical slider range. Data collected on the vertical sliders include the timestamp and input buttons of the keyboard presses, timestamps of alerts and timestamps of misses. If the correct keyboard button is not pressed when a slider is alert (missed alert), it will eventually time out and the triangle will move back in bounds and the alert state is returned to normal. A miss is then recorded in the data.

1.3.7 Communications (Visual) Task

The visual Communications Task is contained in the lower-left quadrant. It contains a set of four channels, labeled OPS1, OPS2, INT1, INT2, together with their current state values (see Figure 7). Above these is a cue box which alerts the participant of changes to be made to the communications channels. When the text in the cue box turns red, it is notifying the participant of a change to be made; additionally, the cue contains a channel to be adjusted and the new target state value for that channel. The participant responds to the alert by using the up/down arrows to select the target channel (selection indicated by a bounding box) and using the right/left arrows to adjust the state value up/down, respectively. Once the target channel state is at the cued value, the participant submits the changes by pressing the "Enter" or "Return" key.

Data recorded in this task include all arrow key presses, with timestamps, and the final submitted value with the timestamp for the Enter/Return key press. If the cue box returns to the non-alert state (text changes from red to white) before the participant submits the corrected channel state, then a miss is recorded.

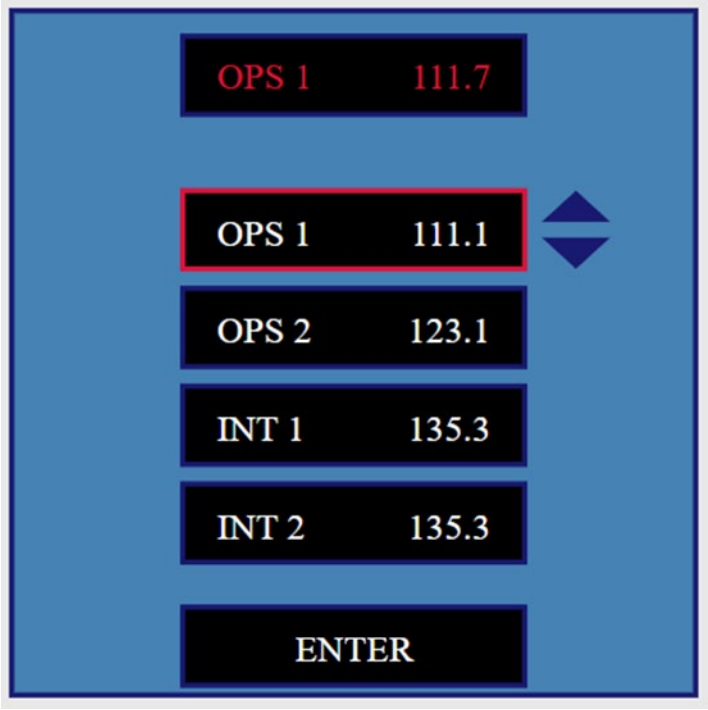


Figure 7. Communications Task

Attributes of the Communications Task that can be modified include the number of channels, ranges of channel values and frequency of alerts. For more information on these attributes, please refer to either Appendix A or the Condition Administration section (5.0).

1.3.8 Resource Management Task

In the resource management task (Figure 8), participants must keep the resource levels in the main tanks as close to the center of the target region as possible. The target region is indicated by the thicker bars on the sides of the main tank boxes. The tanks are connected to a fuel source (circles) and a reserve tank (smaller rectangles). Switches (indicated by the arrows on the connecting lines) act as gates and are available between each main tank and its source and reserves as well as between the two main tanks. Participants can open the gates by selecting the number key associated with each gate in the diagram.

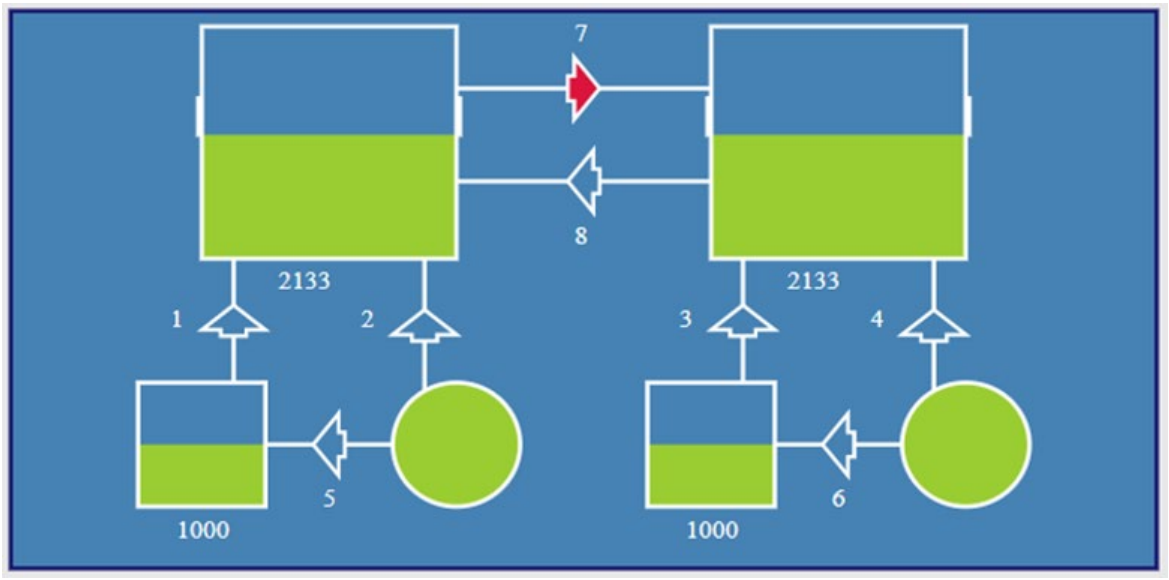


Figure 8. Resource Management Task

Switches can break, as indicated by the arrow turning red. Participants cannot change the state of the broken gates (which eventually time out and return to an off state), but they can adjust the activity of other gates to accommodate any unavailable gates to maintain appropriate resource levels. Participants can interact as much as they want with this task without consequences.

The data recorded in this task consist of each number key press with its timestamp and the current resource levels in the tanks. Further details on the data recorded and configuration options are available in Appendix A or the Condition Administration section (5.0).

1.3.9 Audio Communications Task

An auditory Communications Task is available that can be used as an alternative to the visual Communications Task (Figure 9). For this task, the audio recording will instruct a particular call sign to change the channel to a new frequency, which is called an alert. If the participant's call sign is used, he or she should select the channel, change it to the commanded frequency, and hit "Enter" on the keyboard. The participant will only have a certain amount of time to respond before the alert times out. If the instruction is directed at a call sign different from the participant's, it should be ignored. For more information on these, please refer to Appendix A or the Condition Administration section (5.0).

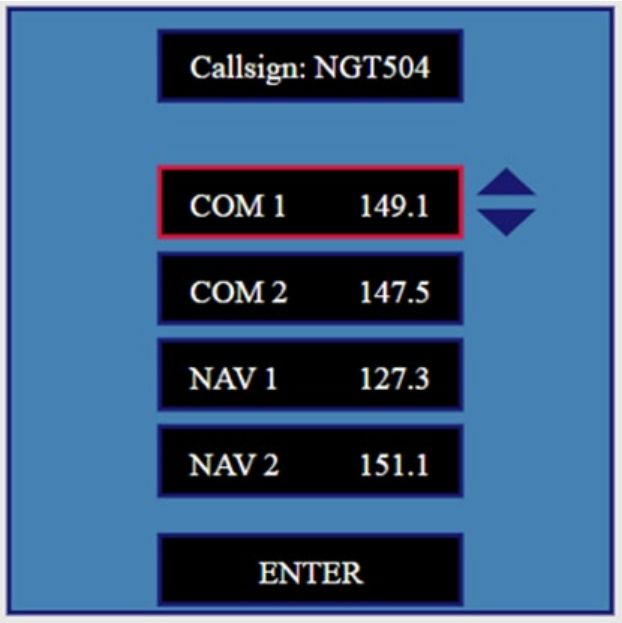


Figure 9. Audio Communication Task

1.3.10 Aircraft Coordination Task

The Aircraft Coordination task was included as a measure of participants' situational awareness. The participants' objective is to keep the incoming plane from colliding with the focal aircraft at the center of the display. Depending on trajectory of the incoming aircraft, the two aircraft may be at risk of colliding. When a second plane appears likely to collide with the focal aircraft, participants must respond to avoid the collision, using either the research-specified keyboard or control buttons. Data logged for this task include the time of aircraft events, time for participant response to incoming aircraft, and the movement of the incoming aircraft on the X- and Y-axes.

There are three visual indicators given to give the user feedback on the appropriate response:

- Participant properly responds and avoids a collision event (i.e., a "hit"; Figure 10A)
    - Center plane will turn green and move vertically to avoid the collision
    - Vertical change is represented by the center plane changing in size
- Participant indicates collision is imminent, but the trajectory of the aircraft did not overlap (i.e., "false alarm"; Figure 10B)
    - Center plane will return yellow and remain in the center of the screen
- Participant fails to respond when a collision event was imminent and the two aircraft collide (i.e., "miss"; Figure 10C)
    - Center plane will collide with incoming plane and turn red

In each of the three scenarios the incoming plane finishes its path across the screen. The color of the center plane will reset when the next incoming aircraft enters the screen.

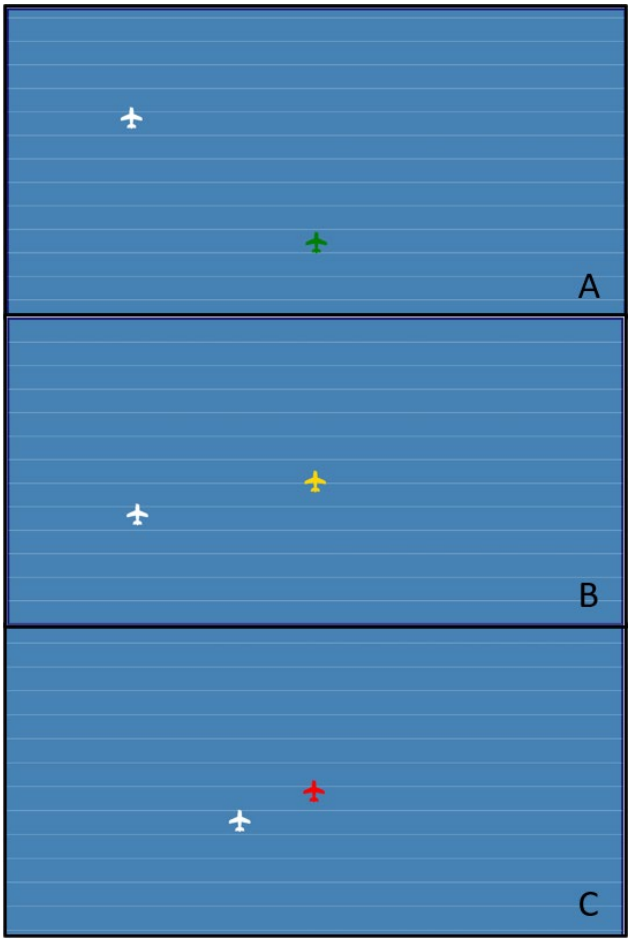


Figure 10. Aircraft Coordination Task with Three Scenarios

1.4 Site Administration Page

The Site Administration page (Figure 11) is where experimental data can be viewed, and conditions can be created or modified. It can be reached by going to host-site/admin/ in a web browser on a client computer. Once there, researchers will be prompted for a username and password, both of which are admin by default. This page is an easy way to confirm that the data from an experiment were correctly saved to the database.

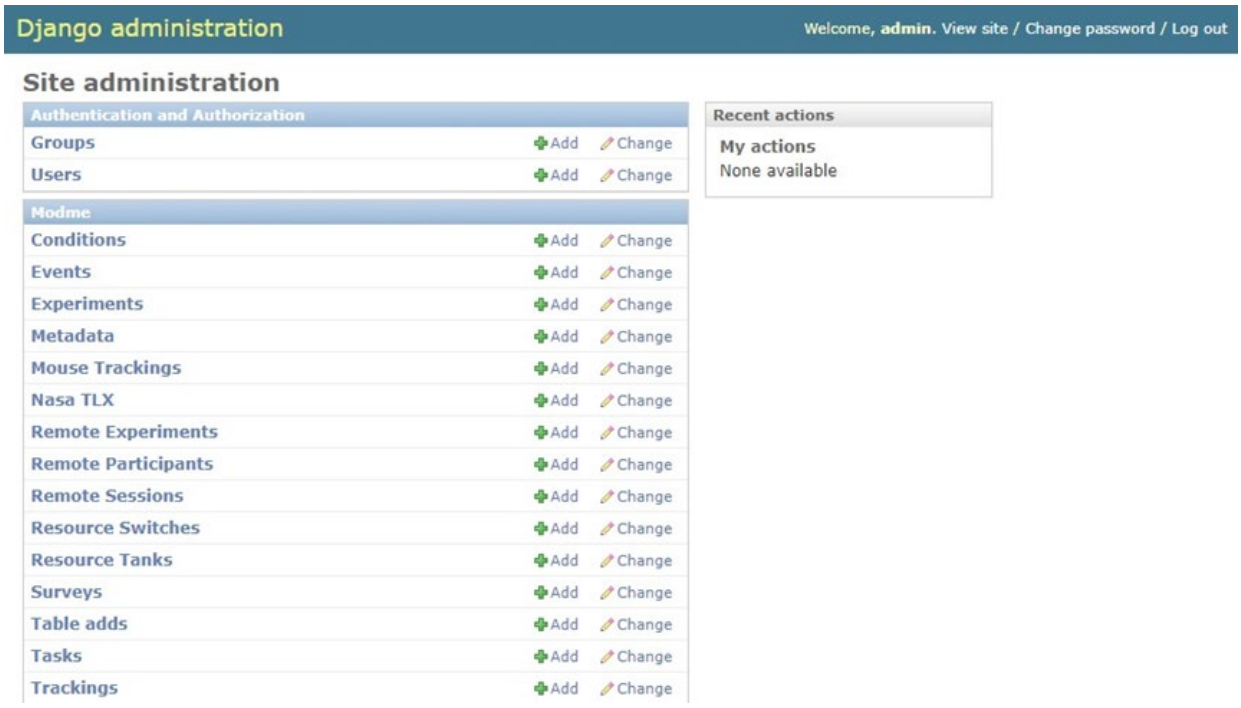
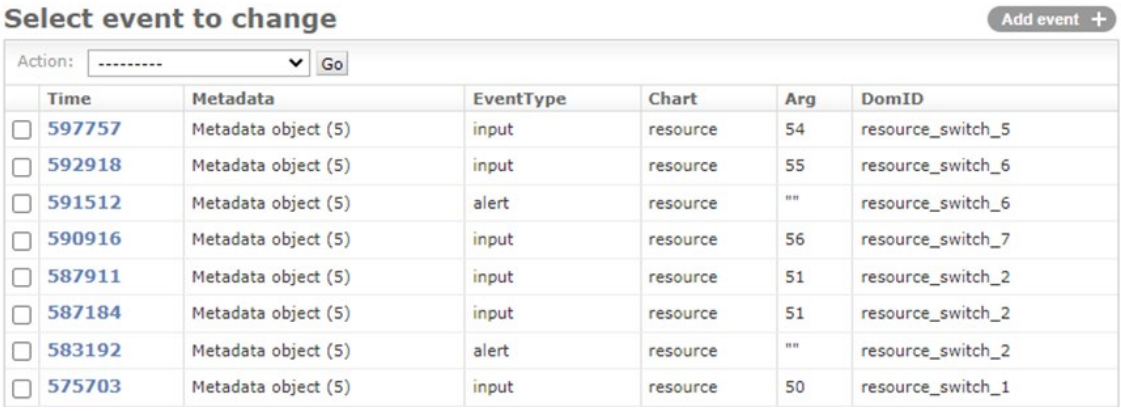


Figure 11. Site Administration Page

Data stored in the database can be seen by selecting the desired table from the list shown. The example below uses the Events table. Once a table is selected, the next to load will provide a preview of the corresponding data entries (Figure 12). Clicking on the time headings listed in the left-hand column for a given entry will direct researchers to a new page with any additional records. Currently, all information is shown in the preview for each data entry.



	Time	Metadata	EventType	Chart	Arg	DomID
☐	597757	Metadata object (5)	input	resource	54	resource_switch_5
☐	592918	Metadata object (5)	input	resource	55	resource_switch_6
☐	591512	Metadata object (5)	alert	resource	""	resource_switch_6
☐	590916	Metadata object (5)	input	resource	56	resource_switch_7
☐	587911	Metadata object (5)	input	resource	51	resource_switch_2
☐	587184	Metadata object (5)	input	resource	51	resource_switch_2
☐	583192	Metadata object (5)	alert	resource	""	resource_switch_2
☐	575703	Metadata object (5)	input	resource	50	resource_switch_1

Figure 12. Event Data Tables

The main purpose of the Site Administration page is creating and editing experimental conditions. When creating a new experimental condition, researchers will be asked to specify a name and duration for the condition. Selecting the “Save and continue editing” button will take then direct users to the experimental condition configuration page. The configuration page allows for easy customization of experimental conditions using a point, click and set approach. To use the page, click on the object to be changed and any properties that can be changed will appear to the left. Changing one of the parameters in the list and then hitting the “Apply” button will update the settings running on the live demo on the right, providing a preview of how the experiment will look when running. Hitting the “Submit changes” button will then save the changes made to the condition to the database. Further details on this process are available in the Experiment Administration section (6.0).

1.5 Experiment Page

The experiment page is where the layout of the screen, collection of desired tasks and parameters can be set and customized. Outlined below are the settings that can be modified or controlled:

Name	Name of the experiment as it appears to participants.
ICD PDF	The informed consent document (ICD).
ICD Confirmation Text	Appears below the ICD, instructing participants to confirm they have read the document, their questions have been answered, and they want to participate in the study.
Skip ICD	Allows the researcher to skip electronic presentation of the ICD if the document is signed on paper or on another electronic form.
Instructional PDF	Appears after the ICD when the experiment begins and should provide a general overview of what participants ought to expect.
Skip Instruction Page	Allows researchers to skip electronic presentation of these instructions if the information is provided in another form.
Instructional Confirmation Text	Appears below the experiment instructions document, asking participants to confirm they have read the instructions and questions have been answered.
Conditions	List of conditions that will be used in the experiment. It is composed to the condition IDs separated by commas.
Randomized Conditions	Option causes the order of the conditions listed below to be presented in a random order.
Experiment Ending Message	Will be shown at the end of the experiment and can be used to provide final instructions, follow-up information, or compensation codes.
Experiment Notes	Notes for experimenters that will not be shown to participants.

1.6 Database

All data are stored in the db.sqlite3 database located in the ModME folder from which the task is running. Tables store the conditions, or the various configurations and parameterizations for conditions developed by the experimenter.

Task-related event and behavior data are stored in the following tables:

- Events
- Metadata
- Mouse Trackings
- NASA TLX
- Resource Switches
- Resource Tanks
- Trackings

2.0 INSTALLATION README

2.1 ModME Overview

The ModME program provides a simple-to-use interface for customizing and running experimental conditions, a collection of human data in the form of inputs to the environment, and a database for organizing experimental parameters and data. ModME is built in Python, JavaScript, HTML, and CSS on top of an SQLite database, allowing for simple storage of experimental conditions and data. This document provides a general introduction to the system requirements and program functions in ModME.

Use the following commands to determine whether the necessary programs are installed.

```
|--|--|--|
|MacPorts|required for Mac| port installation|(https://www.macpor
ts.org/install.php)|
|Python 3.7 or 3.9|required|beginner’s guide|
|pip|required|Recent versions of Python include pip.|
|tar|required|Many alternatives exist, such as 7-zip.|
|virtualenv|optional|Recent versions of Python include virtualenv
      .|
```

Virtualenv allows reseachers to create little VEs, each with its own set of Python packages installed. Using a VE is considered good practice, especially when working with Python, because it prevents programs with different dependencies from interfering with one another.

2.2 Installation Prerequisites

2.2.1 Windows Installation Prerequisites

Individuals with both Python 2 and Python 3 installed should use of Python3 commands to ensures use of Python3, where Python may fall back to Python2. Go to the Python download page and download the latest Python 3.9 or 3.7 installer. Other versions of Python may work, but only versions 3.7 and 3.9 have been verified to work with ModME.

When the download is complete, run the installer and make sure to check the box to add Python to the computer's Path. After the installer finishes, make sure that Python installed successfully. Open a command prompt window and verify the Python version:

```
Python --version
//Or
Python3 --version
//or
py –version
```

If neither version of the command works, then Python was likely installed but not added to the system's Path variable. This can be done manually: Open the Windows main Settings menu and type "Edit the system environment variables" into the search field, then select the name when it appears below the search box. On the screen that then pops-up, click on the button at the bottom labeled "Environment Variables." On the next screen that appears, the field at the top will be labeled "User variables for [desired username]".

In the list of environment variables for the username, click on "Path" and then click the "Edit" button. Click "New" and add the following location (replacing with the desired username): C:_USERNAME

Then, click new and add this other location: C:_USERNAME

Select these locations and click the "Move up" button until they are at the top of the list. Click "OK" to close all of the control panel windows.

The commands above should now work to check the version of Python installed on the computer.

Pip is installed with Python. Installation status and version can be verified with the following command:

```
pip3 -V
```

As with Python, multiple versions of pip may also be installed, each associated with different Python installations. If the wrong version of pip is running, the version associated with the correct version of Python can be specified by entering:

```
Python3 -m pip -V
```

To install *virtalenv*:

```
pip3 install virtualenv
//to verify install
virtualenv --version
```

2.2.2 MacOS Installation Prerequisites (MacPorts)

If both Python 2 and Python 3 have been installed, the best practice is to use the Python3 commands to ensure the system does not default to Python2. There may also have multiple versions of pip, each associated with different Python installations. If the wrong version of pip is running, an alternative is to call the pip associated with a Python install directly:

```
Python3 -m pip -V
```

Note, MacPorts must be installed before running the following installations:

```
sudo port install Python37 py37-virtualenv py37-pip
sudo port select Python3 Python37
sudo port select pip3 pip37
sudo port select virtualenv virtualenv37
```

2.2.3 Linux Installation Prerequisites (Ubuntu 16.04)

As with installation on macOS, researchers installing ModME on a machine running Linux will also start by confirming the version of Python installed. Those with both Python 2 and Python 3 installed should use Python3 commands to ensure use of Python3, as Python may default to Python2. Some users may also have multiple versions of pip, each associated with different Python installations. If the wrong version of pip is running, one solution is to directly call the version of pip associated with the Python installed:

```
sudo apt install Python Python-pip tar
python3 –m pip install virtualenv
```

2.2.4 Verify Prerequisites

Ensure the computer is using Python versions 3.7 or 3.9 and that everything is installed and available from the command line:

```
$ Python3 –version
Python 3.7.7
$ pip3 –version
pip3 20.0.2 from c:-packages(Python 3.7)
$ tar –version
bsdtar 3.3.2 - libarchive 3.3.2 zlib/1.2.5.f-ipp
$ virtualenv –version
virtualenv 20.0.16
```

2.3 Installation

1) Creating a VE to contain the server (optional but recommended). Make sure *virtualenv* is selected on the Path.

```
virtualenv virtmodme
//for Mac and linux
source virtmodme/bin/activate
//for windows
virtmodme\Scripts\activate
```

2) Install Python library dependencies.

```
pip3 install Django==2.2
pip3 install tornado==6.1
```

3) Install and configure the ModME server locally.

```
unzip ~/Downloads/modme.zip
cd modme/
Python3 ./Tornado.py
```

To build the SQL database, use the following code, replacing admin_name email address, password, and download location as appropriate:

```
unzip ~/Downloads/modme.zip
cd modme/
Python3 ./manage.py migrate
Python3 ./manage.py shell -c "from django.contrib.auth.models imp
ort User; User.objects.create_superuser('admin_name', 'admin_name
@example.com', 'admin_password')"
Python3 ./Tornado.py
```

4) To verify the server is running appropriately:

   a. Open a browser to http://localhost:9000

   b. There should be a large banner with the word 'INDEX' and links to ModME, Administration, Conditions and "Add a New Table", as shown in Figure 2.

5) To quit the server, use CONTROL-C in the command window. Then type "exit" followed by pressing the Enter key to exit the VE.

2.3.1 Launching ModME after Installation

1) MacOS and Linux: In the terminal

```
//start virtual environment if it is not already running
 source virtmodme/bin/activate
 cd modme/
 Python3 ./Tornado.py
```

2) Windows: In Command Prompt

```
//start virtual environment if it is not already running
virtmodme\Scripts\activate
cd modme
Python3 ./Tornado.py
```

3) Verify the server is running

   a. Open a browser to http://localhost:9000

   b. There should be a large banner with the word 'INDEX' and links to ModME, Administration, Conditions and "Add a New Table".

2.4 Configuration of a Condition

2.4.1 Create a Condition

1) Open a browser to http://localhost:9000/admin/ModME/condition

2) Enter the username and password created previously to sign in.

3) Click on “Add Condition +”.

4) Enter the name and duration for the experiment, then press “Save”.

2.4.2 Modify the Condition

1) Browse to http://localhost:9000/admin/ModME/condition

2) Find the condition to be modified then click “Configure”.

3) Use the drop-downs under “Included Tasks” to select the tasks for desired experiment.

4) Click on the desired configuration element to open the properties dialog for it.

5) Press “Submit Changes” when done.

2.5 Repeatable Event Sequences

2.5.1 Record a Session

1) Browse to the ModME experiment initiation page http://localhost:9000/ModME/

2) Select a condition from the drop-down, then enter a participant, session, and study id and press the “Begin” button

3) Press the space bar when prompted.

4) Wait for the experiment to finish. It is not necessary to respond to alerts or otherwise while the experiment is running.

5) Once the session is complete, browse to http://localhost:9000/admin/ModME/metadata/

6) Find the entry for the recorded session and click the link.

7) Check the “AllowEventReuse” box and press “Save.”

2.5.2 Record an Experiment Session with a Recordable Sequence of Events

1) Browse to http://localhost:9000/ModME/

2) After selecting the condition from the uppermost drop-down, the “Reuse Session Data” drop-down will be populated.

3) Check the “Reuse Session Data” box and select the corresponding previously-recorded session.

4) Enter the participant, session, and study IDs, then press “Begin”

5) When the participant initiates the session on the next page, the session will reuse the previously-recorded alerts

2.5.3 Modifying a Sequence of Reusable Events

1) Find the ID for the session to be modified (available at http://localhost:9000/admin/ModME/metadata).

2) Grab previously-recorded alerts from the database using the right metadata_id

```
echo $'.mode csv
select time, eventType, chart, domid, arg
  from modme_event
  where metadata_id=6 and eventtype="alert"
  order by id desc
  ;'   | sqlite3 db.sqlite3 > repeatableAlerts.csv
```

3) Modify the alerts in using the preferred program for editing a comma-separated values (CSV) file.

4) Load the alerts using the command-line tools for site management, correcting the condition name as appropriate

```
Python3 manage.py load_events -c demoCondition ./repeatableAlerts
.csv
```

2.6 Recommended Security Changes Before Deploying on a Public Server

These instructions cover the recommended changes to the security settings before deploying ModME on a public server. Generally, the only setting that should be changed is switching the debugging setting to “false”.

Background: ModME comes in debugging mode. In this mode, ModME prints a notable amount of a data into error messages that make debugging errors easier. However, this information can also be used by adversaries to find the best way to attack the server. **It is therefore important to change the debugging setting to “false” before starting a public server.** To do this:

1) Open the file: modme.py

2) Go down to the line that says: “DEBUG = True”

3) Replace “True” with “False”.

4) Save and close file.

3.0 MAC OS AND LINUX INSTALLATION GUIDE

3.1 Installation Overview

This document explains how to set up an instance of a modular multi-tasking environment (ModME) on a machine running either a Macintosh operating system (macOS) or Linux. For more general information about installing and running ModME, please refer to the Installation ReadMe file. This document provides a text-based walkthrough for installation; however, it is intended to be used by individuals with a technical background in computer science. The current is a walkthrough is specific to Macs, though will also work for Linux computers, and is intended for a wider audience.

3.1.1 Installation Prerequisites

Most steps of the installation will involve working in a Terminal window. To open a Terminal window, navigate to the Dock then select Launchpad, then type “Terminal” into the Launchpad search bar at the top. The “Terminal” icon should then appear; click on it to open a Terminal window.

The Terminal window should be similar to Figure 13, though the specific text may differ.

Figure 13. Sample Terminal Window in macOS

ModME currently works with both Python versions 3.7 and 3.9; users may decide which to install. To install Python on a macOS in the Terminal window, enter for following lines, clicking the Enter key at the end of each line:

```
sudo port install python37 py37-virtualenv py37-pip
sudo port select python3 python37
sudo port select pip3 pip37
sudo port select virtualenv virtualenv37
```

To install Python on Linux machines, the port commands are not needed, so enter:

```
sudo apt-get install python3.9 python3-pip
     pip3 install virtualenv
```

An error may appear indicating “Unable to locate package”. This is caused by trying to install an older version of Python. To fix this, more program sources must be added to the same location in which “apt-get” looks for programs. This is done by entering the following command:

```
sudo add-apt-repository ppa:deadsnakes/ppa
```

When prompted, press the Enter key to confirm.

All other instructions throughout this document are identical for both macOS and Linux computers.

3.1.2 Installation Verification

To check that all programs are properly installed, enter the lines listed below (i.e., “You enter”) and the system should respond with the subsequent line (i.e., “System response”). Exact version numbers may vary, but if the lines match otherwise, the correct versions are installed:

You enter:	`python3 –version`
System response:	`Python 3.7.7`
You enter:	`pip3 –version`
System response:	`pip3 20.0.2 from c:-packages(python 3.7)`
You enter:	`tar –version`
System response:	`bsdtar 3.3.2 - libarchive 3.3.2 zlib/1.2.5.f-ipp`
You enter:	`virtualenv –version`
System response:	`virtualenv 20.0.16`

If the system does not generate the desired response, it indicates the corresponding installation may not have executed correctly. In that case, re-enter the installation command lines again. If this does not work, then contact someone experienced with programming and systems management for assistance.

3.2 Installing Python Libraries in Virtual Environments

ModME requires some Python libraries also be installed. Libraries contain code to do commonly needed things and are used to save time during development. Installing the libraries will make them available to Python programs and it is recommended that they be installed on a VE. This will prevent other programs on the computer from being affected by changes made for ModME.

- **The *ls* and *cd* commands**

For users not familiar with command line navigation using the Terminal, there are two important commands to memorize: *ls* and *cd*. To use the Terminal window to access the ModME folder, type "ls" into the command line and then press the enter key. The *ls* command stands for "list" and using it will list all files or sub-directories within the current directory (i.e., folder). The current directory will be shown in the Terminal window on the active command line, to the left of where new text is entered. In most cases, this will start in the "Home" directory, similar to what is shown in Figure 14:

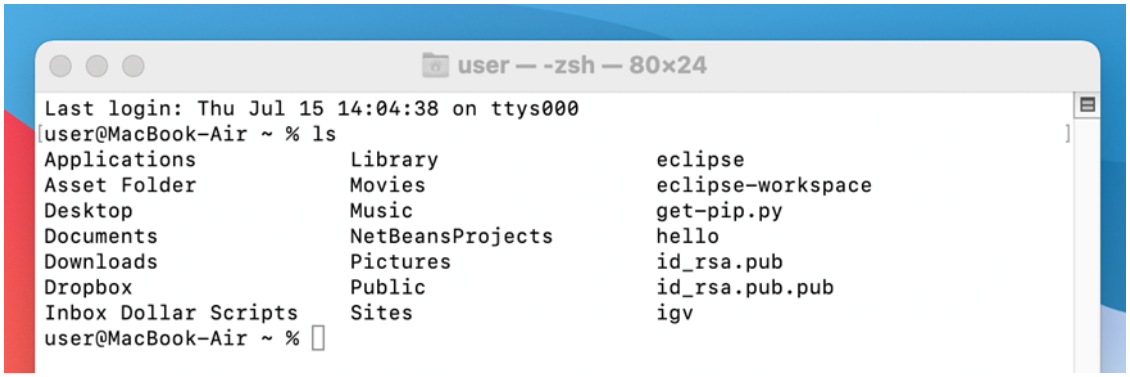


Figure 14. Sample Terminal Window when Installing Python Libraries on macOS

The command “cd” stands for “Change Directory”. It is used to move between directories in the terminal by entering the path to the desired directory. For example, using the “ls” command in the example above will produce a list of available sub-directories, and the list should include a sub-directory named “ModME”. To make “ModME” the current directory, enter in the command “cd ModME” then press the enter key. The location shown in the Terminal will then change to the ModME directory (Figure 15).

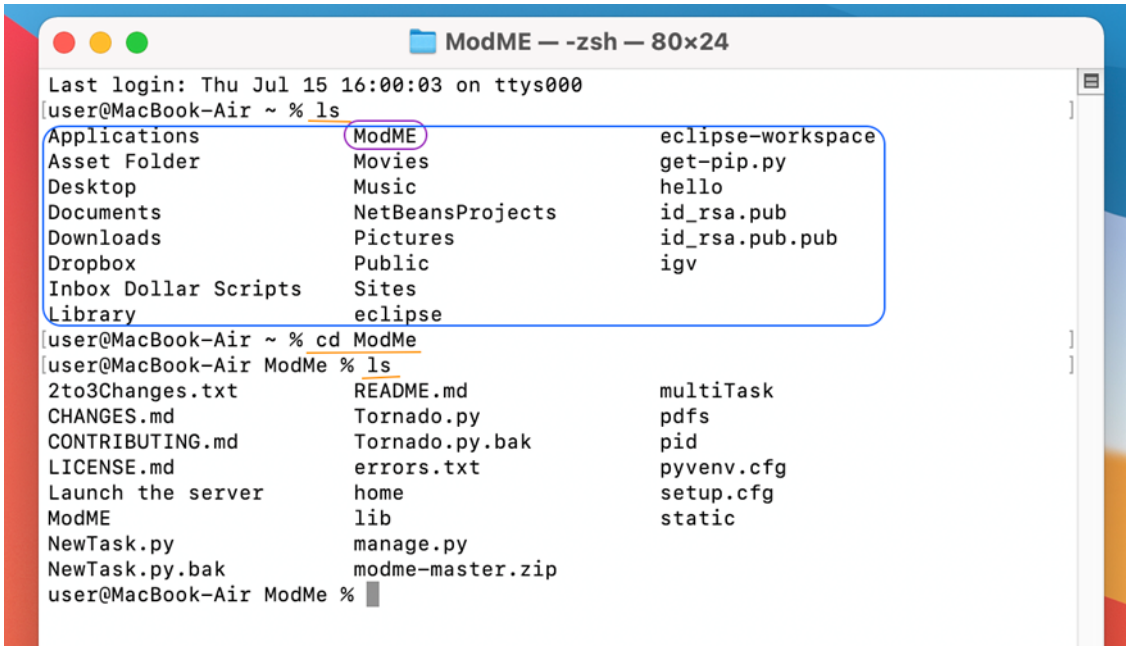


Figure 15. Example of Changed Directory

3.2.1 Creating a Virtual Environment

The VE is created with the “virtualenv” command. To use “virtualenv”, first enter the following command to create a VE. The second part of the command assigns a name to the VE; in this example it is named “nameOfVirtualEnv”:

```
virtualenv nameOfVirtualEnv
```

The VE can be assigned any desired name by replacing “nameOfVirtualEnv” in the command above, but for simplicity the name “nameOfVirtualEnv” will be used throughout the remainder of this guide.

To activate the VE, enter the following into the Terminal command line:

```
source nameOfVirtualEnv/bin/activate
```

The assigned name of the VE will now be shown before the current address line.

With the VE activated, libraries installed with Pip will be installed to the VE without impacting other programs on the computer.

3.2.2 Installing Libraries

ModMe requires two libraries to be installed with Pip: Django and Tornado. To install these libraries and their dependencies, enter the following commands:

```
pip3 install Django==2.2
pip3 install tornado==6.1
```

To see a list of installed libraries, enter the following command:

```
pip3 freeze
```

3.2.3 Starting Tornado Service

Using the Terminal window, navigate to get to the ModME folder using the commands given above. Save the ModME folder to a location on the computer where it will be easy to locate. The desktop is easy to navigate to from the Home directory, which makes saving the ModME folder simple. If saved to the desktop, enter the following commands to get to the ModME folder:

```
cd Desktop
cd modme
```

Once inside the ModME folder, type the "ls" command to display a screen similar to Figure 16. The list will include a file called "Tornado.py", which confirms the correct directory is active.

```
user@MacBook-Air ModMe % ls
2to3Changes.txt         README.md               multiTask
CHANGES.md              Tornado.py              pdfs
CONTRIBUTING.md         Tornado.py.bak          pid
LICENSE.md              errors.txt              pyvenv.cfg
Launch the server       home                    setup.cfg
ModME                   lib                     static
NewTask.py              manage.py
NewTask.py.bak          modme-master.zip
user@MacBook-Air ModMe % %
```

Figure 16. Example Use of ls Command

To proceed, enter one of the following commands into the Terminal window:

```
Python3 ./Tornado.py
Python ./Tornado.py
```

The “Python3 ./Tornado.py” command should be entered first, but may result in an error due to the machine’s current operating system. If that happens, then enter the “Python ./Tornado.py” command. If both result in an error, either Python or Tornado is not properly installed. The problem may be resolved by revisiting the installation instructions and/or contacting an experienced programmer team for assistance.

If either of the commands work, the subsequent message should resemble Figure 17:

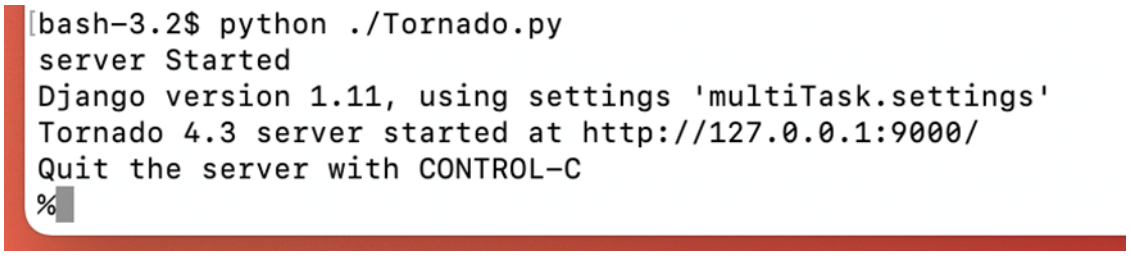


Figure 17. Example After Python Installation

Once installation of Python and Tornado are confirmed, open a web browser (Chrome, Firefox, or Safari are recommended) and enter “localhost:9000” into the address bar (Figure 18). This will load the ModME Index page, which includes links to the Administration and ModME experiment pages. Both the default username and password are “admin.” For instructions on changing either the username or password, please refer to the Installation ReadMe section of this document (Section 2).

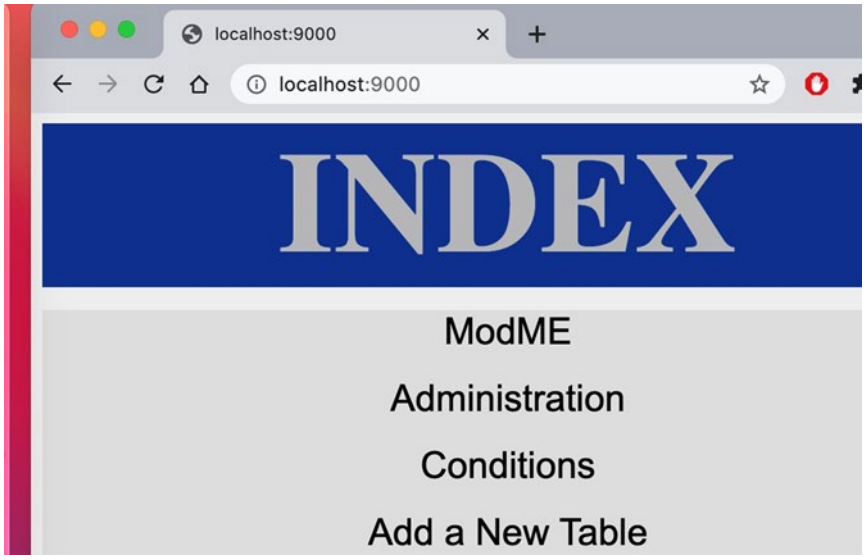


Figure 18. Localhost:9000 Address and ModME Index Page

3.2.4 Stopping Tornado Service

Once Tornado service is started, it will continue to run in the background and must be stopped manually. This will also stop ModME until it is manually restarted and will save all admin and session data to the database file. To stop the Tornado service, open a Terminal window and press the Control and “C” keys on the keyboard.

4.0 WINDOWS INSTALLATION GUIDE

This document explains how to set up an instance of a modular multi-tasking environment (ModME) on a computer running a Windows operating system.

For more general questions and steps, the Installation ReadMe (section 2.0) is a great reference. This provides a text-based walkthrough for installation; however, it is intended for use by those who have extensive experience with application modification and development. The current guide is specific to Windows computers and is intended for a wider audience.

4.1 Installation Prerequisites

4.1.1 Python

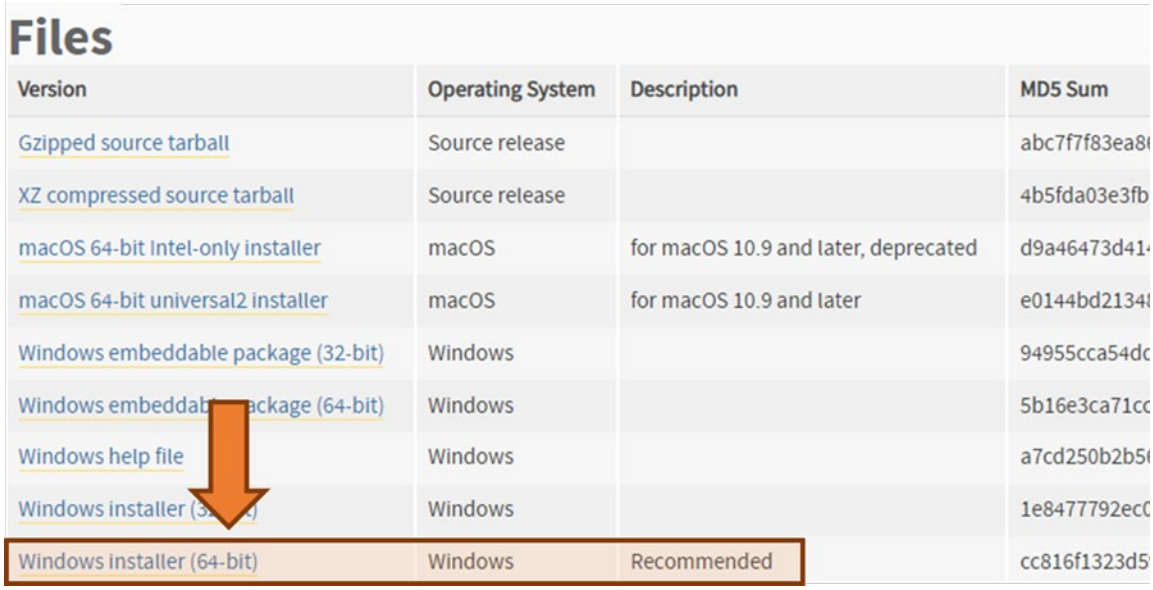
The first step is to install Python on the computer. If Python is already installed, proceed to the section “Creating a Virtual Environment” (section 4.1.3).

The easiest way to install Python is to access https://www.python.org/downloads/ and navigate to the download page for the latest release version of Python 3.9.X, under the section labeled “Looking for a specific release?” Select the Download link that corresponds to the most recent version, which should be near the top (see Figure 19 below).

Note that there may be more recent releases than Python 3.9 (e.g., Python 3.10.4 in the sample image), but ModME is only guaranteed to work on a computer running Python 3.9.

Figure 19. Recent Python Releases

Scroll to the bottom of the screen to find the section labeled “Files” (Figure 20). From this list, click on the version named “Windows installer (64-bit)”. A window will appear confirming the program is being downloaded.



Files

Version	Operating System	Description	MD5 Sum
Gzipped source tarball	Source release		abc7f7f83ea8
XZ compressed source tarball	Source release		4b5fda03e3fb
macOS 64-bit Intel-only installer	macOS	for macOS 10.9 and later, deprecated	d9a46473d41
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	e0144bd2134
Windows embeddable package (32-bit)	Windows		94955cca54dc
Windows embeddable package (64-bit)	Windows		5b16e3ca71cc
Windows help file	Windows		a7cd250b2b5
Windows installer (32-bit)	Windows		1e8477792ec
Windows installer (64-bit)	Windows	Recommended	cc816f1323d5

Figure 20. Windows Download Selection

Open the computer's Downloads folder and locate an executable file called "python-3.9.X-amd64.exe". Right-click on the file name and select the option "Open" to start the automatic installation process. The only modification that will need to be made to the default settings is on the "Advanced Options" screen. As shown in Figure 21, the box labeled "Add Python to environment variables" must be checked. Note that Pip is automatically installed as part of Python, so no additional step is needed for this program.

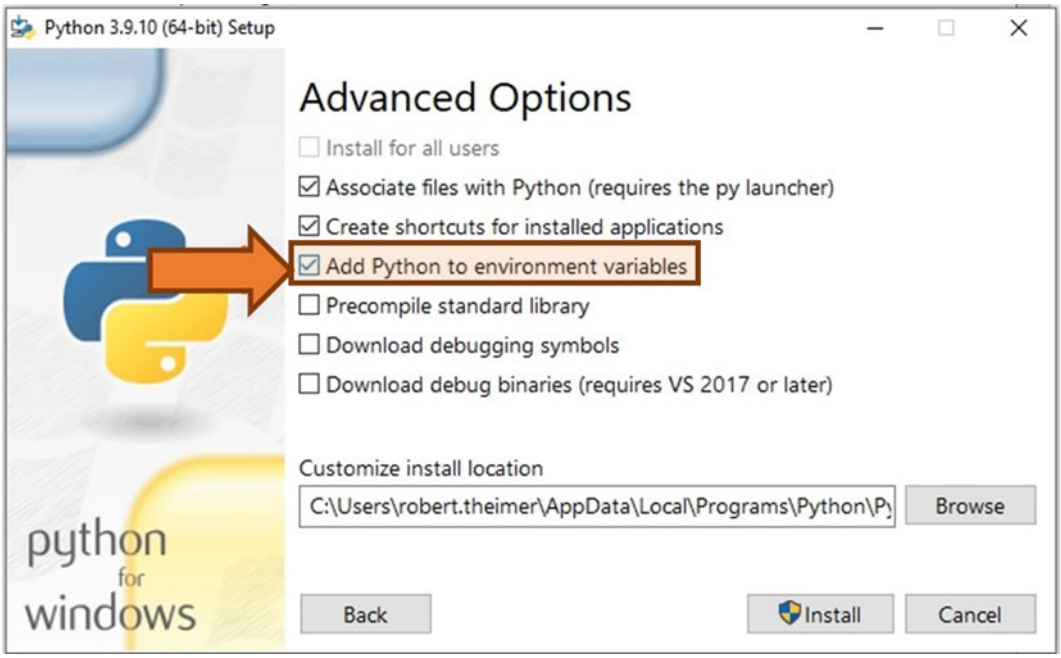


Figure 21. Advanced Options when Installing Python

Once Python has been installed, the next step will be to install the virtual environment (VE), or *virtualenv*. A VE is a program that allows the user to create self-contained packages of Python dependencies. This is important for preventing Python programs from interfering with each other.

This step will require entering code directly into the Command Prompt. To open the prompt, navigate to the Start menu and open the item labeled "Command Prompt", or enter the string "cmd" into the Windows search field (Figure 22).

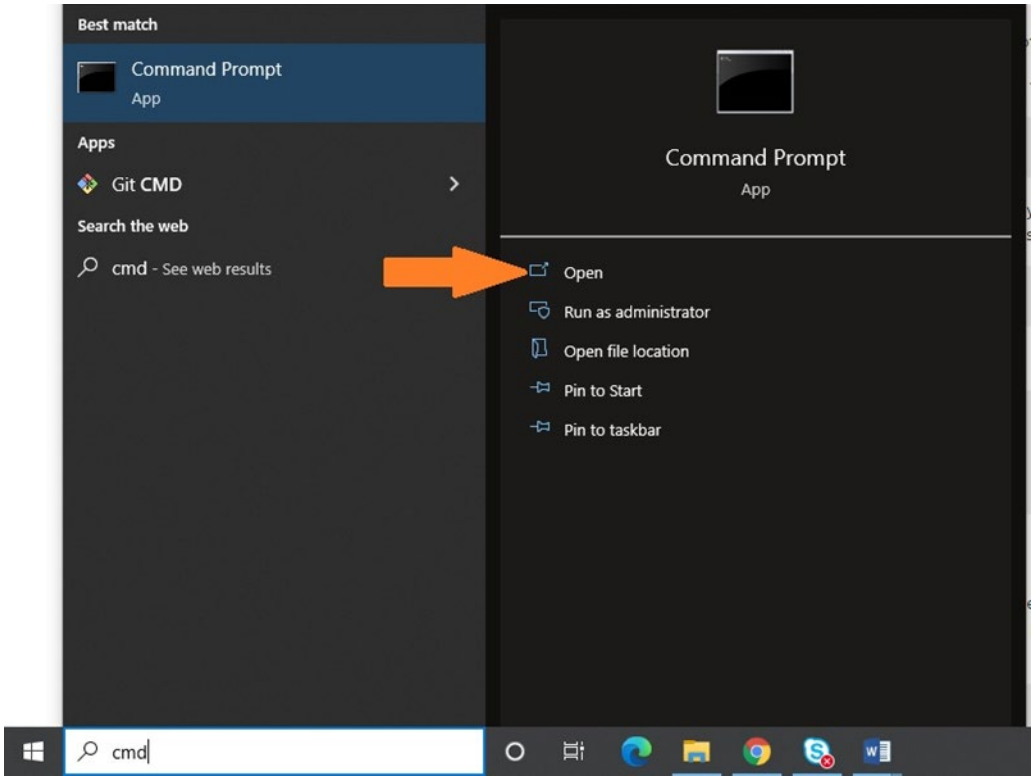


Figure 22. Command Prompt Tool

On the first open line of the Command Prompt window, enter the following command:

```
pip3 install virtualenv
```

This will install the VE and any other necessary packages (Figure 23). If an error message appears after entering the command, try the steps again. If the problem continues, please contact the systems administrator associated with your organization for additional support.

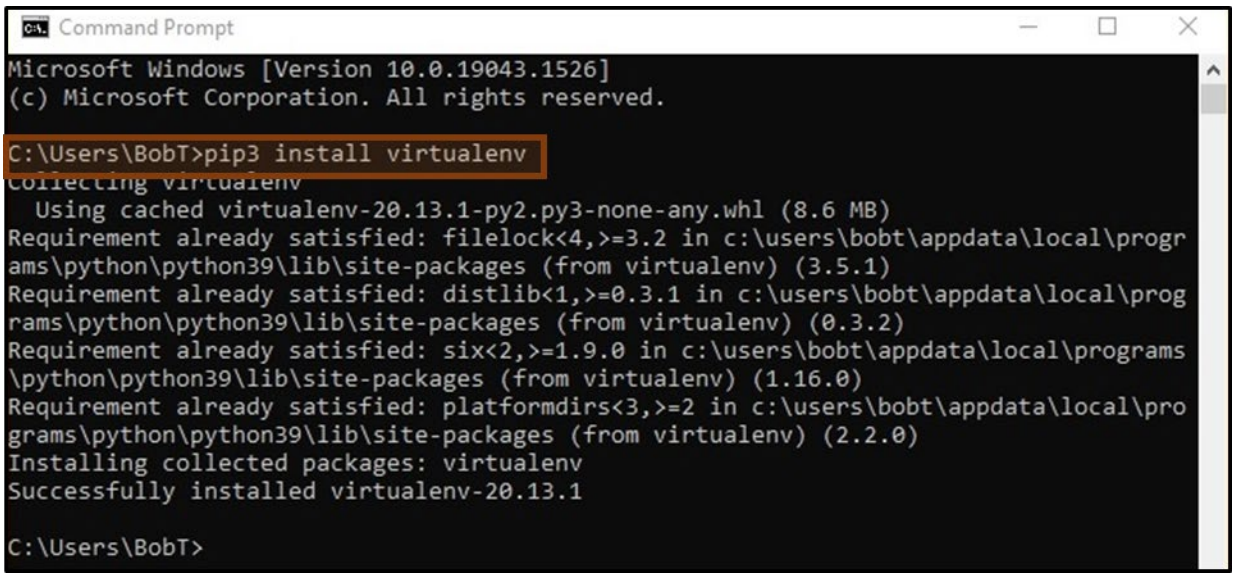


Figure 23. Command to Install VE

4.1.2 Installation Verification

After the VE has been installed, the final step in this phase is to verify that both Python and the VE are working properly. To do this, enter the commands listed below, hitting the "Enter" key after each.

```
py --version
pip3 --version
virtualenv --version
```

The exact folder names and pathways will differ between users and machines, but the results will be similar to Figure 24.

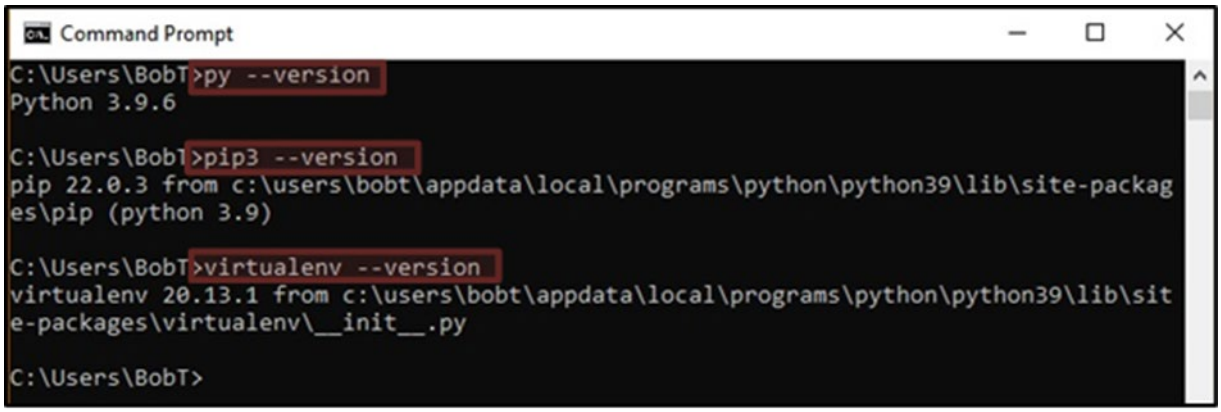


Figure 24. Commands to Verify Python and the VE have been Installed Properly

Note: it is possible to have multiple versions of Python and Pip installed. If the output for the above commands is not the same as or similar to that shown in Figure 24, it is possible that older versions of Python and/or Pip are installed on the machine. If that is the case, the following commands should be used to call the latest versions of both programs.

```
python3 --version
```

```
py --version
```

4.1.3 Creating a Virtual Environment

As mentioned above, creating a VE to run ModME and contain server dependencies will improve the functionality of the program. VEs are created using the command “virtualenv”. The command to create a VE involves two parts, the actual command and the name assigned to the environment. The code is entered as:

```
virtualenv nameOfVirtualEnv
```

For the example in Figure 25, the VE has been named “virtModME”, indicated with the brown outline.

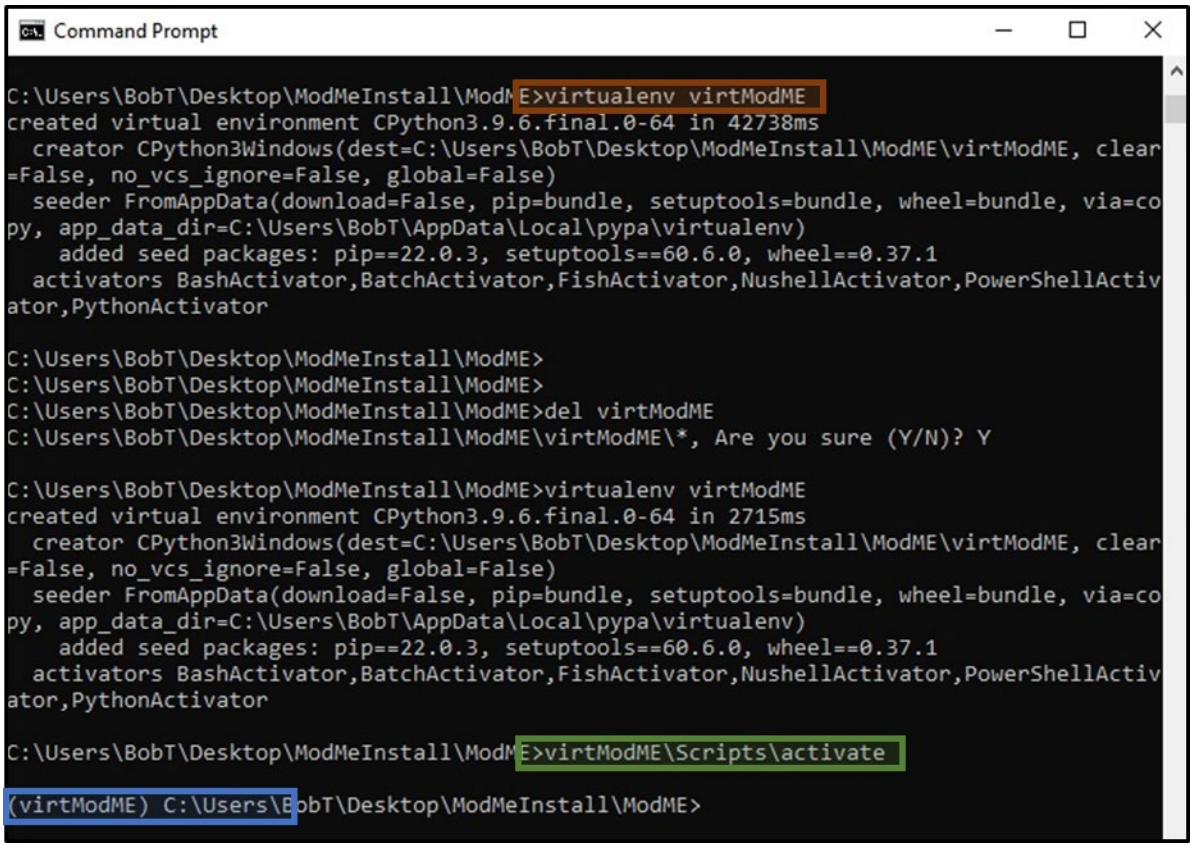


Figure 25. Commands Used to Create and Activate the VE

To activate the VE, enter the following command (see Figure 25, green outline):

```
nameOfVirtualEnv\Scripts\activate
```

The name of the VE will appear before the address on the command line (see Figure 25, blue outline).

4.1.4 Installing Dependencies

With the VE activated, libraries installed with Pip will be installed to the VE without impacting other programs on the same computer. ModME requires two libraries to be installed with Pip: Django and Tornado.

To install these libraries and their dependencies, enter the following commands (Figure 26, brown outlined segments):

```
pip3 install Django==2.2
pip3 install tornado==6.1
```

The list of the installed libraries can then be called using the command below, as indicated with the green box outline in Figure 26, green outline:

```
pip3 freeze
```

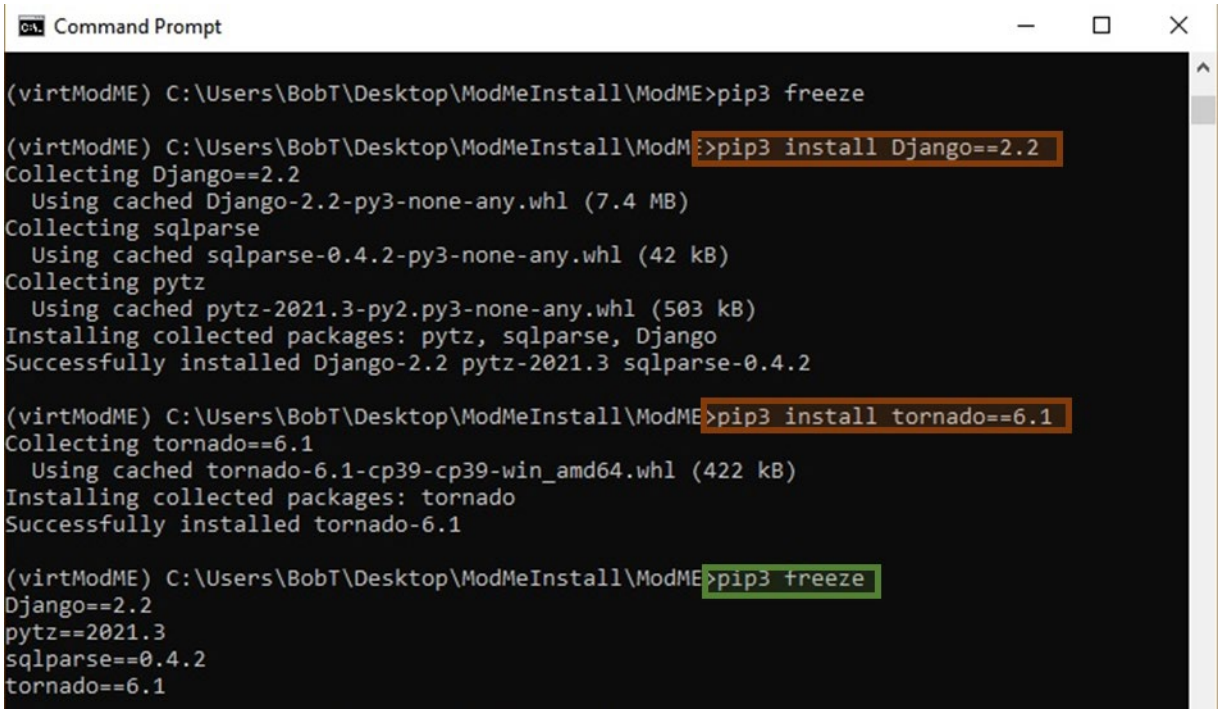


Figure 26. Commands to Install Django and Tornado and Show Associated Libraries

4.1.5 Creating the ModME Database

ModME runs on a SQLite database engine. Inclusion of the library in the ModME installation folder may depend on how the program was obtained (e.g., downloaded from GitHub, installed from a disc, etc.). If the ModME installation folder includes a file called “db.sqlite3”, no further steps are necessary and the computer should be ready to run ModME properly.

If the “db.sqlite3” file is not in the installation folder, the next step will be to build the database. This is done by entering the following command:

```
python3 ./manage.py migrate
```

The next step is to create the Administrative (i.e., “Admin”) account by entering command below into the Command Prompt window (Figure 27). Note that the name (“admin_name”), email address (admin_name@example.com), and password (“admin_password”) should be modified. This can be done through the code included below, and an example is included in Figure 27:

```
python3 ./manage.py shell -c "from django.contrib.auth.models imp
ort User; User.objects.create_superuser('admin_name', 'admin_name
@example.com', 'admin_password')"
```

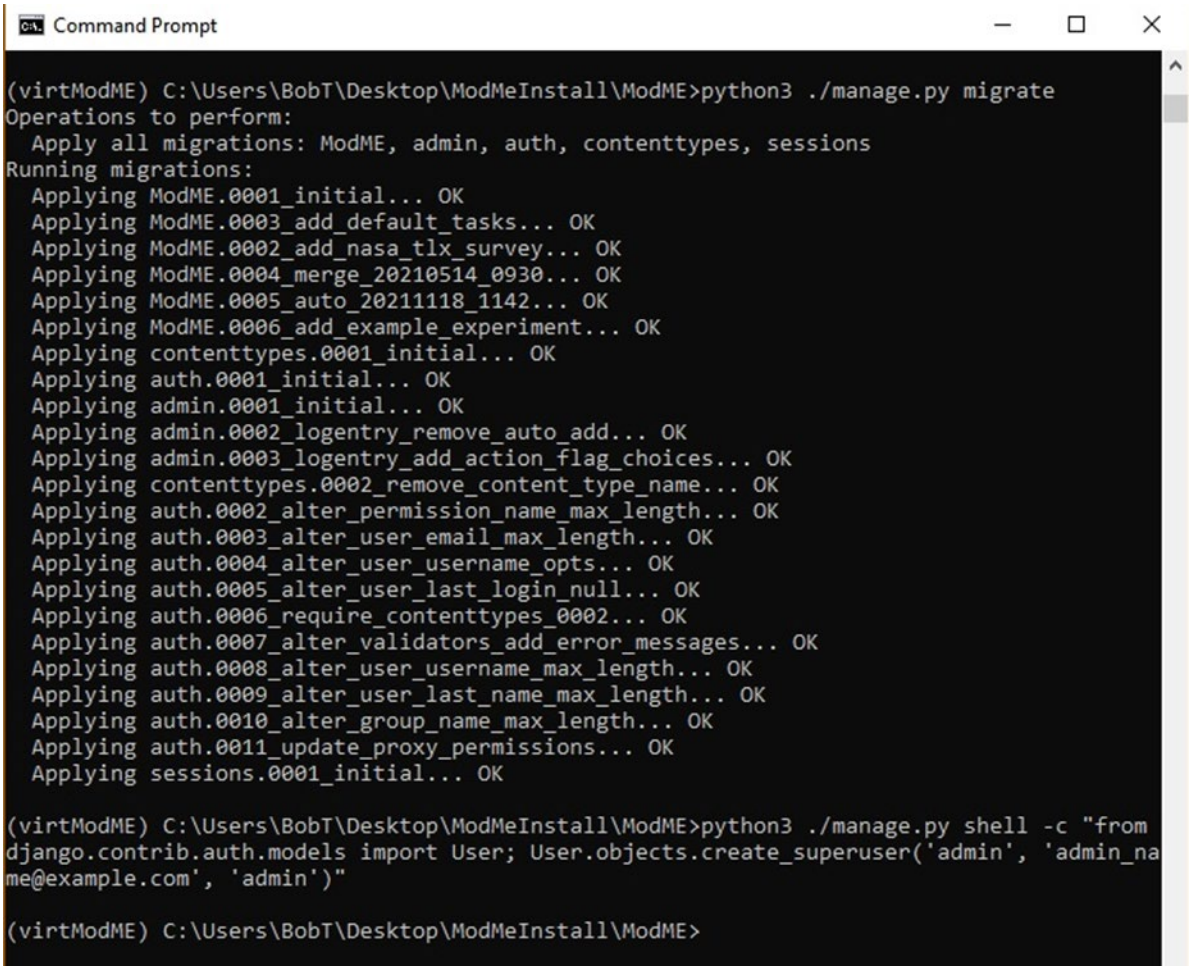


Figure 27. Creation of Admin Account

4.2 Running and Stopping ModME

Running ModME starts by initiating the Tornado Service, which is done with the following command (Figure 28):

```
python3 Tornado.py
```

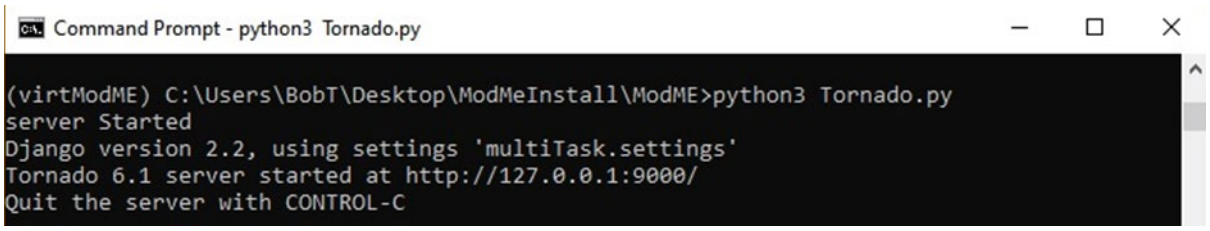


Figure 28. ModME Starts by Initializing Tornado

To access the server, open a web browser and enter http://localhost:9000 into the address field. This will display the ModME home page shown in Figure 18.

To stop the ModME server, enter control + c into the Command Prompt window. If that does not work, refresh the browser page.

5.0 CONDITION ADMINISTRATION

5.1 Overview

The purpose of this document is to walk experimenters through the process of creating, configuring, and administering conditions and experimental sessions using the Modular Multi-tasking Environment (ModME). In this context, "condition" refers to a set of tasks running concurrently within an experimental session. Condition settings include task configuration and specifications, such as the number and type of tasks included. Conditions are saved in ModME and can be used across all experiments created with the same account. An experimental session may include a single or multiple conditions, according to the desired difficulty or duration of a study.

Figure 29 shows the main Conditions administration page where researchers can create and manage conditions for use in ModME experiments. There are nine components to this page:

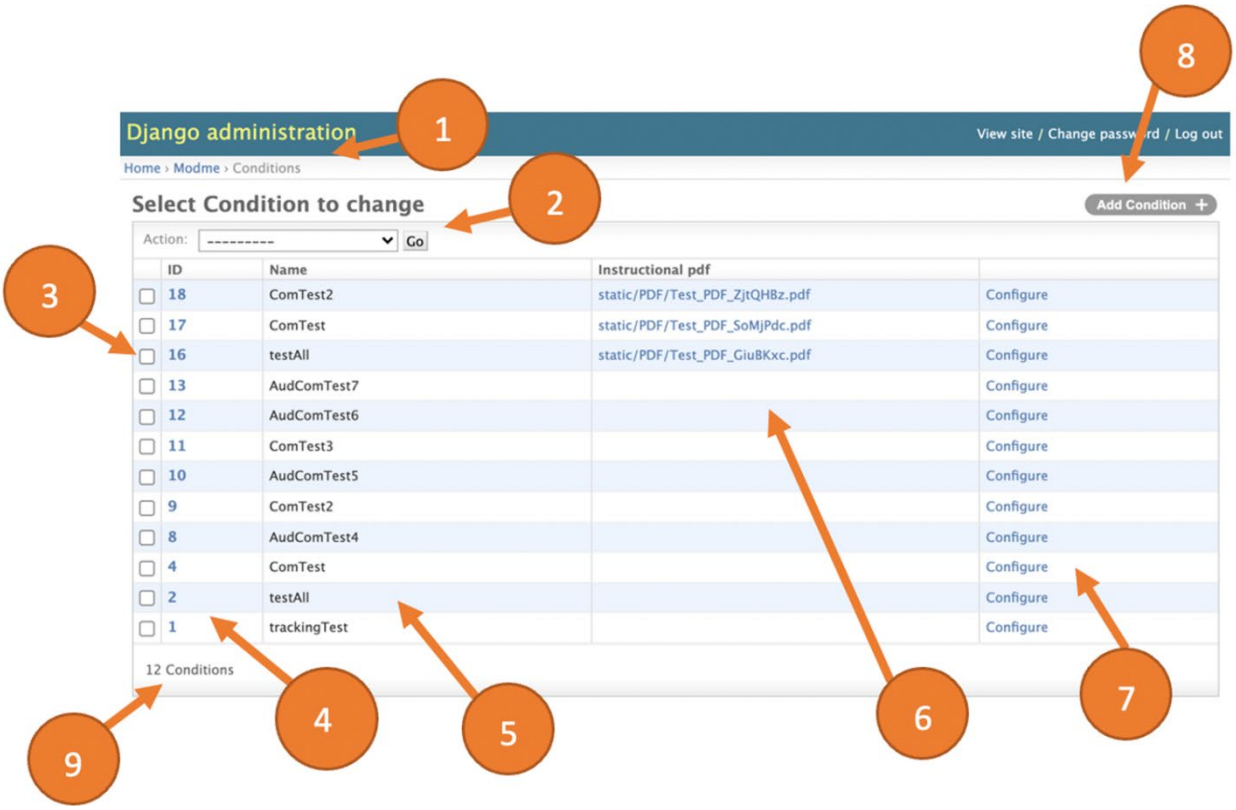


Figure 29. Sample Administration Condition Page

1) **Location:** Indicates current page while logged into the Administration Site. In the screen shot above, the current location is Home > Modme > Conditions.

2) **Action Selection:** Drop-down field from which to select "actions" that may be taken on the listed conditions. To perform an action, select a condition in the Action Item Selection Column, select the desired action, and then click on the button labeled *Go*. A confirmation pop-up will appear on the screen asking for confirmation, at which point the requested action will be completed.

3) **Action Item Selection Column**: These are the check boxes in the left-most column. Multiple conditions will need to be selected in this column before accessing the *Action Selection* tool described previously.

4) **Condition ID Column:** Lists the IDs of each condition created in the account.

5) **Name Column:** Lists the name assigned to each condition.

6) **Instructional PDF Column**: This is the full file path and name of the instructional document linked to a condition. Note, these are not instructions for completing the individual tasks but rather are instructions specific to the specified condition and used to explain each step in greater detail to participants.

7) **Configuration Column**: Direct links to each of the existing conditions where modifications can be made or verified.

8) **Add Condition Button:** Click this button to create a new condition. Refer to Section 5.2 below for additional detail on the process of creating a new condition.

9) **Instance Counter**: Indicates how many conditions are currently in the corresponding ModME account.

5.2 Creating a New Condition

After clicking on the "Add Condition" button or selecting the ID number associated with a condition, the form shown in Figure 30 will appear.

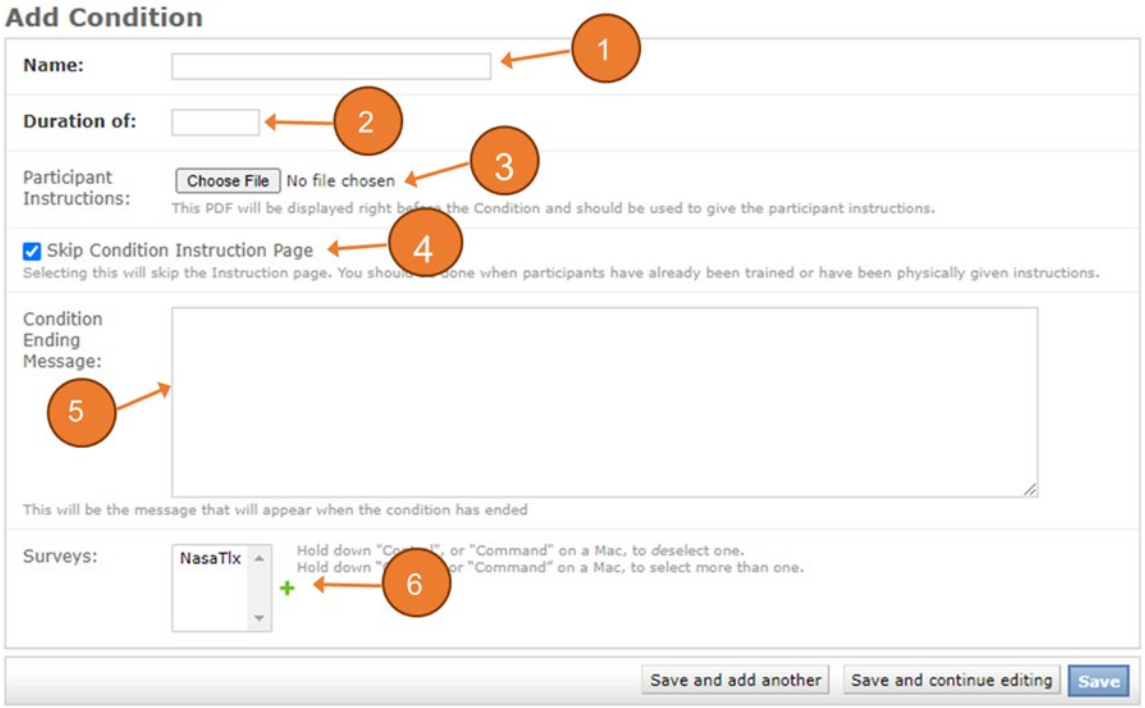


Figure 30. Sample Form to Add Condition

1) **Name:** Enter a meaningful name for the condition. Any alphanumeric text may be entered here. Names are most effective when they are meaningful, distinct, and short.

2) **Duration of**: This is the desired length of time the condition will run in minutes. For instance, if "5" is entered in this field, the condition will run for 5 minutes.

3) **Participant Instructions:** Instructions for the condition, saved as a PDF, to be presented at the start of the condition. This step may be omitted if not relevant; see "Skip Condition Instruction Page" below.

4) **Skip Condition Instruction Page:** If selected, the instructions for the condition will not be presented to participants, even if a file has been attached

5) **Condition Ending Message:** This field is included to present a message to participants after completing the task condition. Possible messages may include telling participants to take a 5-minute break or to tell them the study is complete.

6) **Surveys:** This is used to append survey at the end of the condition. The National Aeronautics and Space Administration Task Load Index (NASA-TLX) has already been built into the system, but other surveys may be added as needed.

5.3 Configuration

Once a new condition has been created, it must be assigned attributes (i.e., duration, included tasks) before it can be used in an experiment. This process is referred to as configuring a condition and can be initiated from the main Conditions page. From here, click on the Configure link to the right of the condition to be modified. In Figure 31, the condition "ComTest" is selected for configuration.

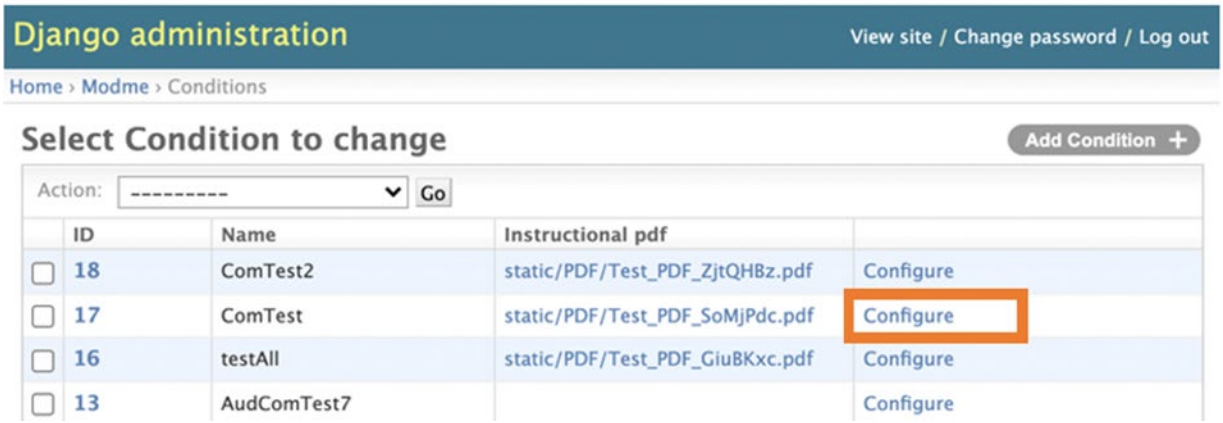


Figure 31. Condition Page Indicating Link to Configure a Condition

5.3.1 Condition Configuration Page

After clicking on the Configure link, the form shown in Figure 32 will appear on the screen. On this screen, the desired tasks can be selected and the duration for the condition can be set. If the condition has not yet been configured, items 1 through 4 will be blank because the corresponding tasks have not yet been selected.

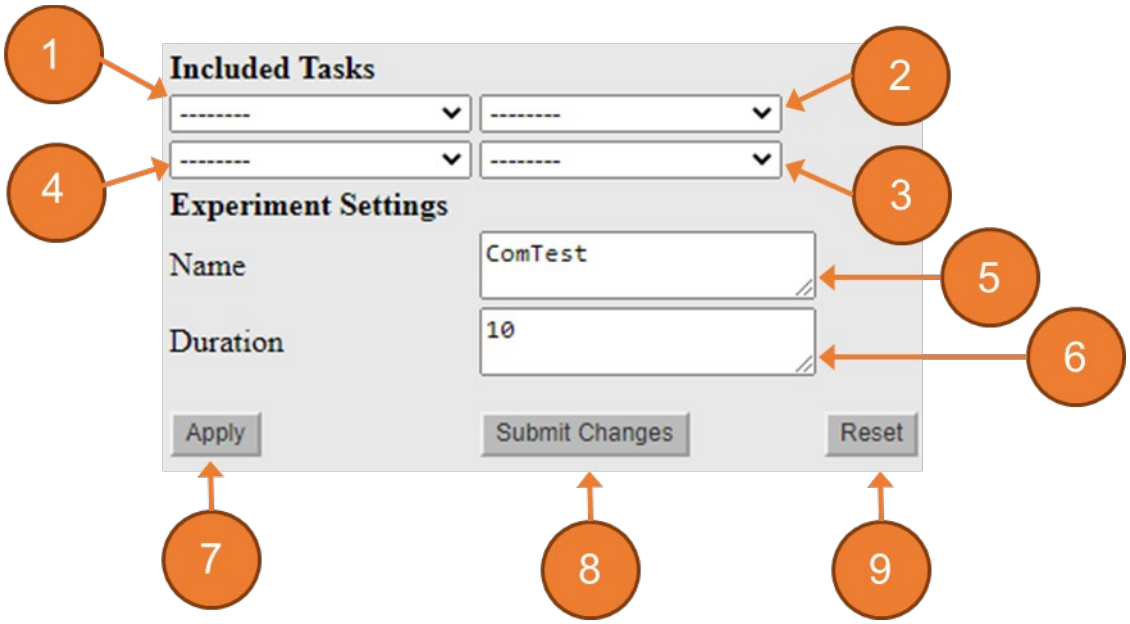


Figure 32. Blank Form for Configuring New Condition

1) **Task 1:** Field to select which task will appear in the top left quadrant.
2) **Task 2:** Field to select which task will appear in the top right quadrant.
3) **Task 3:** Field to select which task will appear in the bottom right quadrant.
4) **Task 4:** Field to select which task will appear in the bottom left quadrant.
5) **Name:** Name of the condition is automatically filled in to match the name of the condition selected on the previous page.
6) **Duration:** The desired number of minutes the condition will run. Note, to optimize data storage, conditions should be no longer than 20 minutes. The default duration for all conditions is 10 minutes.
7) **Apply:** Clicking this button applies changes made to configuration settings for the condition.
8) **Submit Changes:** Saves the configuration changes made and returns the participant to the main Conditions page.
9) **Reset:** Reverts changes made to the condition.

The sample screen shot captured in Figure 33 now includes four separate tasks assigned to condition ComTest. Reviewing the tasks clockwise, starting from the top left quadrant, the tasks are:

1) **Task 1:** Audio Communication task
2) **Task 2:** Tracking task
3) **Task 3:** Resource task
4) **Task 4:** Crosshair Tracking task

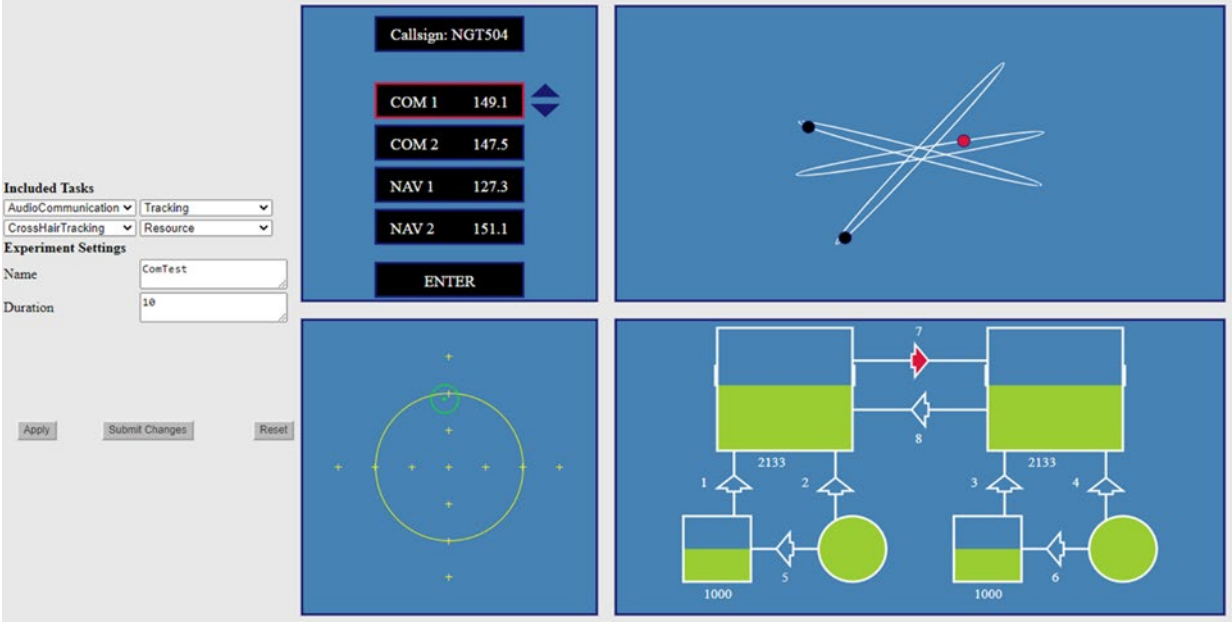


Figure 33. Configuration Page with Tasks Selected and Assigned to Each Quadrant

5.3.2 Condition Tasks

When configuring a new or existing condition, researchers will first need to determine which tasks they want to include and the desired orientation. Figure 34 provides a preview of the selection screen, with the Communication task already selected to be in the upper right quadrant of the condition window. If no task has been selected for a given quadrant, the drop-down field will appear as "- - - - - - - -". The selection of other tasks is explained below.

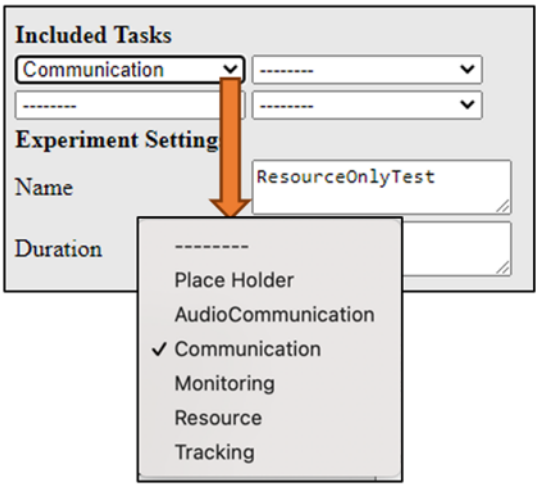


Figure 34. Fields Available to Select Tasks to Include in Condition

5.3.3 Place Holder

The second option in the drop-down task list is a place holder. When a condition does not include four tasks to populate all four quadrants, tasks may expand to fit the screen, but selecting the place holder option prevents this from happening. In Figure 35, the sample condition on the left has two tasks selected for the top two quadrants with "place holder" selected for the bottom quadrants. The condition on the right selected the same two tasks on the top, but left the task for the bottom quadrants undefined (i.e., - - - - - - - -).

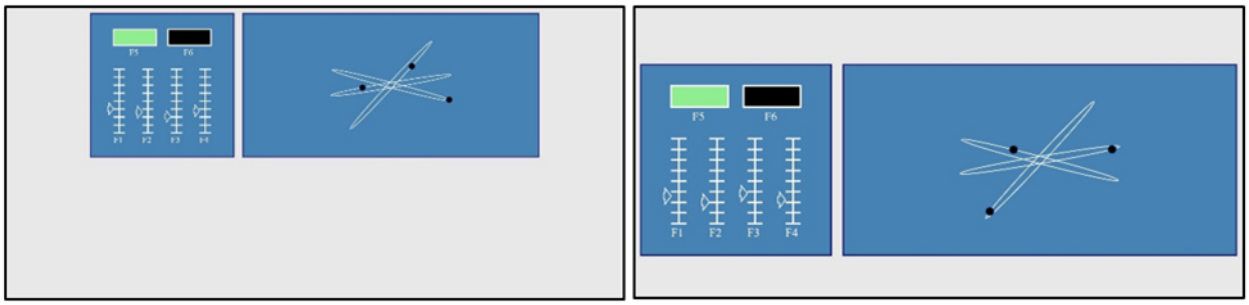


Figure 35. Comparison of condition Screens with (Left) and without (Right) Place Holder

5.3.4 Audio Communication Task

For this task (Figure 36), a recording will play instructing a call sign to change a channel to a new frequency, which is called an alert. Participants only need to respond if their designated call sign is used, clicking the "ENTER" button after changing to the specified frequency. Researchers can control the number of seconds participants have to respond by changing the value in the "Alert Timeout" field within the "Audio Communication" configuration options.

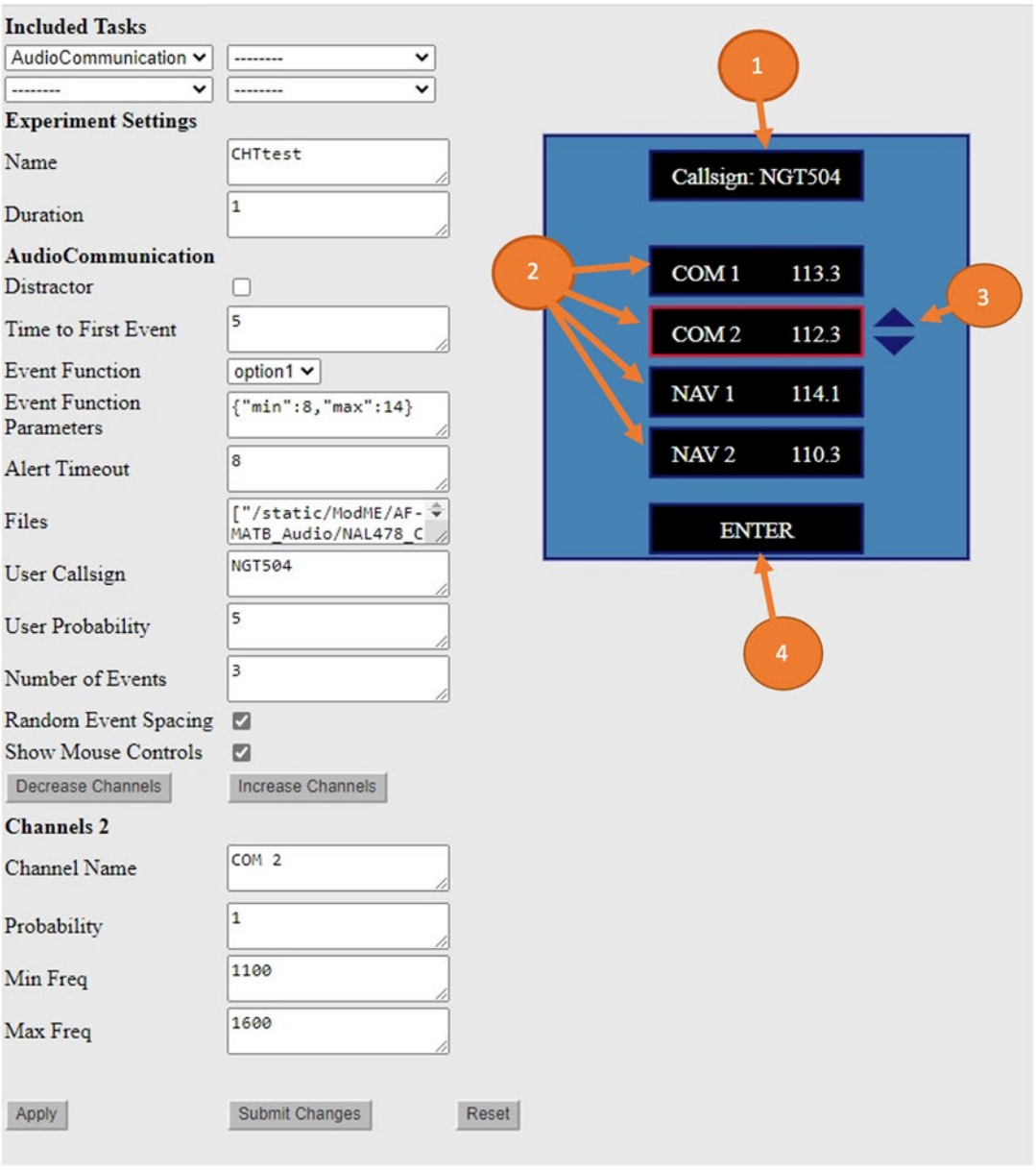


Figure 36. Configuration Options for Audio Communication Task

The Audio Communication Task consists of:

1) **Callsign Field:** This box displays the participant's call sign.
2) **Communication Channels:** A set of boxes, each with a name and current frequency. The participant can select different channels using the up and down arrow keys or by clicking on a channel. When the box is selected, the box's outline will be red.
3) **Frequency Change Arrows:** The participant can change the frequency of the current box by using these arrows or with the left and right arrow keys.
4) **ENTER:** Once participants have changed the frequency, they must confirm the change by clicking on this button or pressing the "Enter" key on the keyboard. After a change has been submitted, the outline of the box will turn green until the participant needs to make another change.

There are several customization options available to researchers:

- **Distractor:** Checking this box turns the task into a distractor. This means that audio command will use the participant's call sign and subsequently never require input from the participant. The task will only be there to provide audio and visual stimulus.
- **Time to First Event:** Sets the number of seconds after the start of the condition until the first audio command is given.
- **Event Function:** Controls how audio commands are dispersed.
    - Option 1 – Alerts will be presented at random intervals, within the established alert and event function parameters min and max.
    - Option 2 – Alerts will be presented at an interval set by the researcher using the event function parameter *avg_wait*.
- **Event Function Parameters:** This box contains the parameters used by the event function. If the random function (option 1) is selected, then the min and max times will be needed. If the geometric function is selected, then the *avg_wait* will be needed.
- **Alert Timeout:** Determines the number of seconds the participant has to respond before the instruction times out.
- **Files:** This box contains the audio list of files that the task will play as commands. The files should have names of the form Call sign_Channel_Freq.wav. If the call sign in the filename is equal to the participant call sign setting below, then the command will be considered one that the participant should respond to. Other commands should be ignored by the participant.
- **User Callsign:** The callsign assigned to participants, which is also displayed in the text box at the top of the task. It may also be used to select which of audio files belong to the participant.

- **User Probability:** This is the probability (out of 10) that an instruction with the participant's call sign will be called. For example, if "4" is entered, then four out of 10 instructions will be addressed to the participant's callsign, while the remainder will be addressed to fictional call signs.
- **Number of Events:** This setting is used if the Random Event Spacing box below is not checked. It will evenly space the alerts such that the desired number of alerts will be presented during the task.
- **Random Event Spacing:** If this is checked, then the event function parameter will be used to distribute the alert; otherwise tasks will be evenly spaced to match the Number of Event setting.
- **Show Mouse Controls:** If checked, the up and down arrows will display next to the selected channel. The arrows can be clicked to adjust the channel's frequency up or down.
- **Decrease Channels / Increase Channels:** These buttons are used to control the number of channels included in the condition.

Additional settings can be specified for each individual channel by selecting the desired channel on the configuration page:

- **Channel Name:** Displayed on the left side of the channel's box and is used to determine if the correct channel was selected after an alert.
- **Probability:** Controls the probability of the specified channel being called in relation to the other channels included in the condition. Using the example in Figure 36, if four channels are included in a condition, and the probability assigned to each condition is "1", then each channel will have a probability of 1/4 or 25% of being specified when the participants' callsign is used. However, if the probability of "COM 2" is changed to "3", then that channel will have a probability of 2/5 or 40%, and the probability of the other channels will decrease to 1/5 (20%).
- **Min Freq:** The minimum starting frequency of the channel.
- **Max Freq:** The maximum starting frequency of the channel.

5.3.5 Communication Task

This task is operationally similar to the Audio Communication task but relies only on visual stimuli and includes fewer configuration options (Figure 37). Periodically, text in the instruction box (1) will turn red to indicate an alert. When this happens, participants should select the specified channel and set its frequency to match what is shown in the instruction box, then click on the "ENTER" button to submit the change. If the submitted change is correct, the text of the instruction box will turn white again; otherwise, the instructions will stay red. If the participant does not submit the change within a specified amount of time, the alert will timeout and the instruction box text will turn back to white.

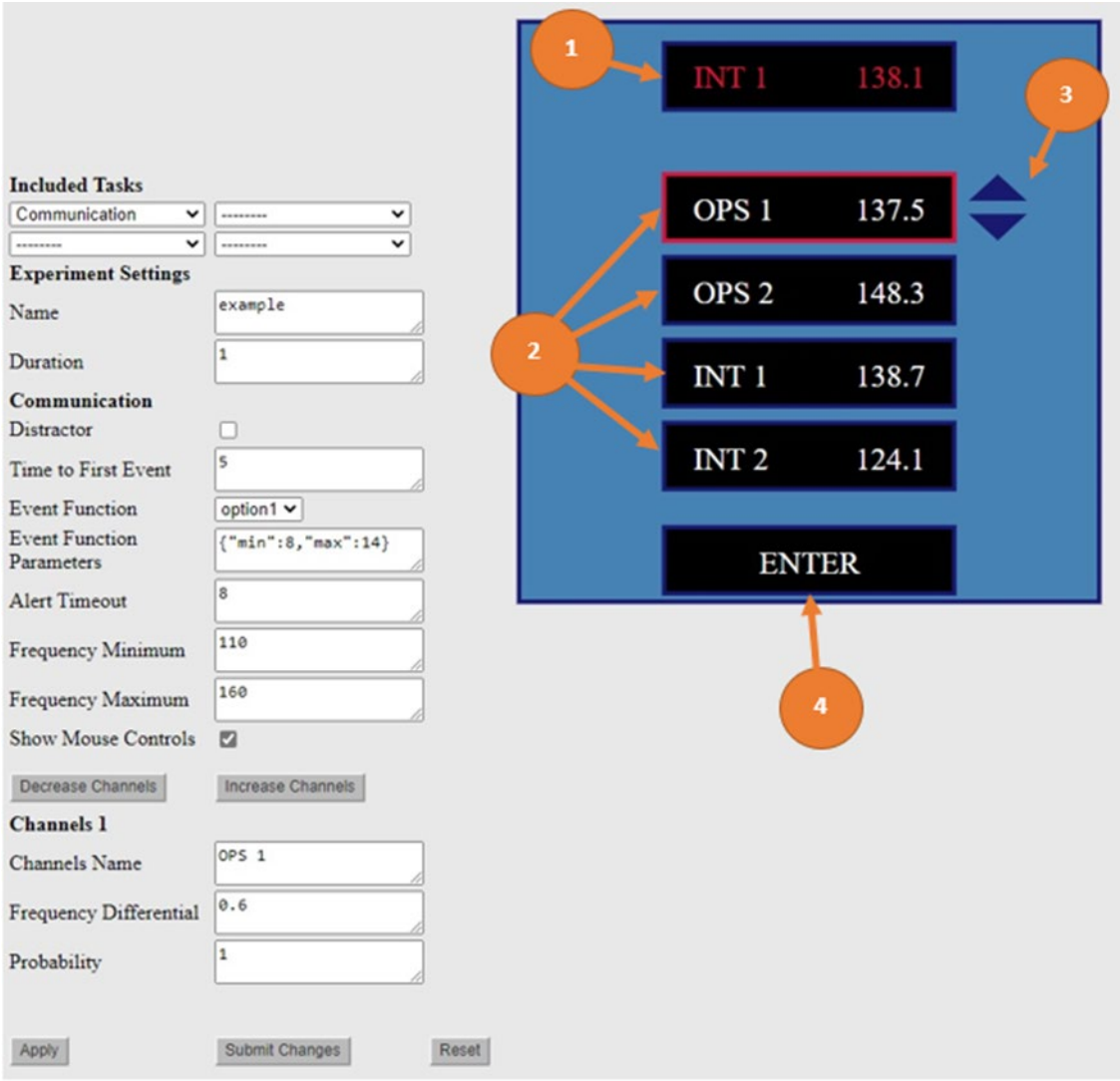


Figure 37. Configuration Options for Communication Task

The Communication Task consists of:

1) **Instruction Box:** This box displays a channel and the target frequency.
2) **Communication Channels:** These are the channels included in the condition, with the current frequency setting specified on the right side of the field.
3) **Frequency Change Arrows:** Participants can control the frequency of the selected channel using either these arrows on the screen, or the left and right arrow keys on a keyboard.
4) **ENTER:** Once participants have changed the frequency, they must confirm the change by clicking on this button or pressing the "Enter" key on the keyboard.

There are a variety of settings researchers can use to modify this task:

- **Distractor:** Checking this box turns the task into a distractor, meaning that the instruction text will never turn red, and the participant will never have to interact with the task. The task will only be included in the condition to serve as visual stimulus.
- **Time to First Event:** Sets the number of seconds after the start of the task until the first alert is given.
- **Event Function:** This controls how alerts will be presented to participants.
    - Option 1 – Alerts will be presented at random intervals, within the established alert and event function parameters min and max.
    - Option 2 – Alerts will be presented at an interval set by the researcher using the event function parameter *avg_wait*.
- **Event Function Parameters:** This determines contains the parameters used by the event function. If option 1 is selected there will be a random amount of time between alerts within the *min* and *max* event function parameters. If option 2 is selected, then the alerts will be geometrically distributed with an average time set by the event function parameter *avg_wait*.
- **Alert Timeout:** Determines the number of seconds the participant has to respond before the instruction times out.
- **Frequency Minimum:** The lowest frequency that will be chosen for an alert event.
- **Frequency Maximum:** The maximum frequency that will be chosen for an alert event.
- **Show Mouse Controls:** If checked, the up and down arrows will be displayed next to the selected channel.
- **Decrease Channels / Increase Channels:** These controls are used to configure the number of channels included in the condition.

There are also settings that can be adjusted to each channel specifically. These can be found by clicking on the desired channel in the configuration page:

- **Channels Name:** This is displayed on the left side of the channel's box.
- **Frequency Differential:** The max difference that an alert can be from the current frequency of a channel.
- **Probability:** The probability that the channel will be chosen. The probabilities of all the channels are cumulative. For example, if there are four channels with probabilities of "1", then the probability of any channel being selected is 1/4. However, if the probability of one of the channels is changed to "2", then the probability of that channel being selected is 2/5 and the probability of the other channels is 1/5.

5.3.6 Crosshair Tracking Task

The objective of the Crosshair Tracking task is to keep the target (Figure 38, label 1) inside the target area (label 2) using a mouse or gamepad controller. Using the mouse, participants can click on the target and drag it. To use the gamepad, participants will hold a button on the controller and move the joystick to control the target.

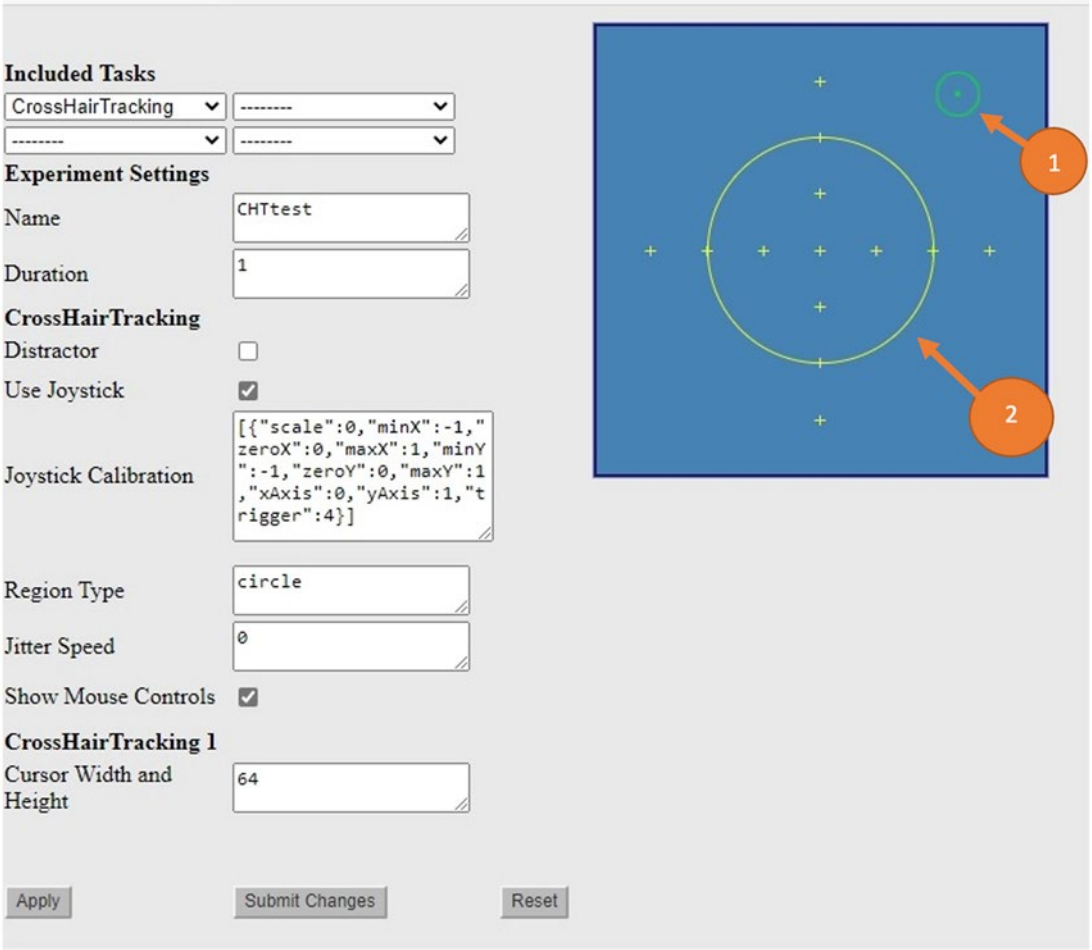


Figure 38. Configuration Options for Crosshair Tracking Task

1. **Target:** This moves around randomly. It can be moved by a joystick, if configured, or by clicking and dragging using the mouse. When the target is clicked on or selected in the joystick it will turn red.
2. **Target Region:** This is a circle or rectangle. The participant's goal is to keep the target inside of this region, and the program logs when the target moves beyond the boundary.

Listed below are the settings that can be configured. These can be viewed by clicking the target in the configuration page:

- **Distractor:** Checking this box turns the task into a distractor. This means that the target will never exit the region and so the task will never require input from the participant. The task will still be included as a visual stimulus.
- **Use Joystick:** If the box is not checked, then the participant may only control the target with the mouse; gamepad cannot be used. When this box is checked, it allows the participant to control the target with the gamepad by holding down a button on the controller and moving joystick. If the box is checked, participants may still click on and drag the target with the mouse.

5.3.6.1 Joystick Calibration

Additional configuration settings are available when using a gamepad controller. These settings will not be applied if the "Use Joystick" option is not selected. There are no configuration options specific to mouse input.

- **scale:** Controls joystick sensitivity, and the value specified correlates positively to joystick sensitivity; higher values translate to greater target movement with a given amount of joystick input. A value of "0" will prevent the target from moving in response to movement in response to joystick input
- **xAxis, yAxis:** Determines which axis on the controller can be used to move the target. *xAxis* specifies what axis move the target left and right, and *yAxis* specifies what axis move it up and down. Tools are available online to identify the most appropriate settings for a given controller (e.g., https://gamepad-tester.com/).
- **trigger:** Determines which button on the controller needs to be pressed to control the target. Mapping options can be identified by using an external gamepad tester (e.g., https://gamepad-tester.com/).

The other configuration settings available will typically not need to be adjusted but modifying the controls may be useful in certain circumstances.

- **zeroX, zeroY:** Sometimes a controller may need to be calibrated, such as when movement is necessary to hold the target in place. If this happens consistently, *zeroX* and *zeroY* should be used to control what joystick position equates to "0" for the corresponding axis.

- **minX, maxX, minY, maxY:** If the controller is more sensitive in one direction, then these settings are used. A normal controller range is from -1 to 1 on each axis, with 0 corresponding to no input, but some controller may only go from -0.7 to 0.8. These settings may be modified to offset such discrepancies.
- **Region Type:** This controls the shape of the target region, either circle or rectangle. If any other setting is entered, the task will default to using a rectangle. Please note that the region dimensions may look different during an actual experimental session (e.g., a circle may look more like an oval).
- **Jitter Speed:** This controls how fast the target randomly moves. Larger values correspond to faster movements, and a value of "0" will keep the target from moving. This can be useful when calibrating controller settings.
- **Cursor Width and Height:** Used to control the size of the target.

Other settings can be modified by changing settings saved in the SQL database. These allow for more exact control over the appearance of the task and are documented here for researchers who are comfortable manually editing the database.

- **rangeX, rangeY:** These settings allow for precise control of the tick marks and target region placement in a manner that reduces distortion potentially caused by screen dimensions. By default, *rangeX* and *rangeY* are "8", meaning the box is 8 units horizontally and vertically, creating the coordinate system that the other appearance settings can use. For example, coordinate (0, 0) is the top left corner, and the center of the field would be (4, 4) since this is midway along the x- and y-axes.
- **targetRegion:** This is an array that only contains one object, "radius", and controls the radius of the target region. The region will always be centered in the task box. If Region Type is set to "circle", then a circle will be drawn with this radius; otherwise, a rectangle will be drawn with the appropriate side radius units from the center.
- **crossHairMarks:** An array that contains the location of each tick mark. Each of the *crossHairMarks* objects contains a *locX* and *locY* which are the number of units from the top-left corner of the task box with the units defined by the *rangeX* and *rangeY* settings.

5.3.7 Monitoring Task

The goal of the task is to scan the gauges and respond to any alerts. Alerts are indicated when either a button (label 1 in Figure 39) changes color or one of the sliders (label 2) moves beyond the target range. The task objective is to reset the object (i.e., button or slider icon) within the desired amount of time, after which the object will return to its base state.

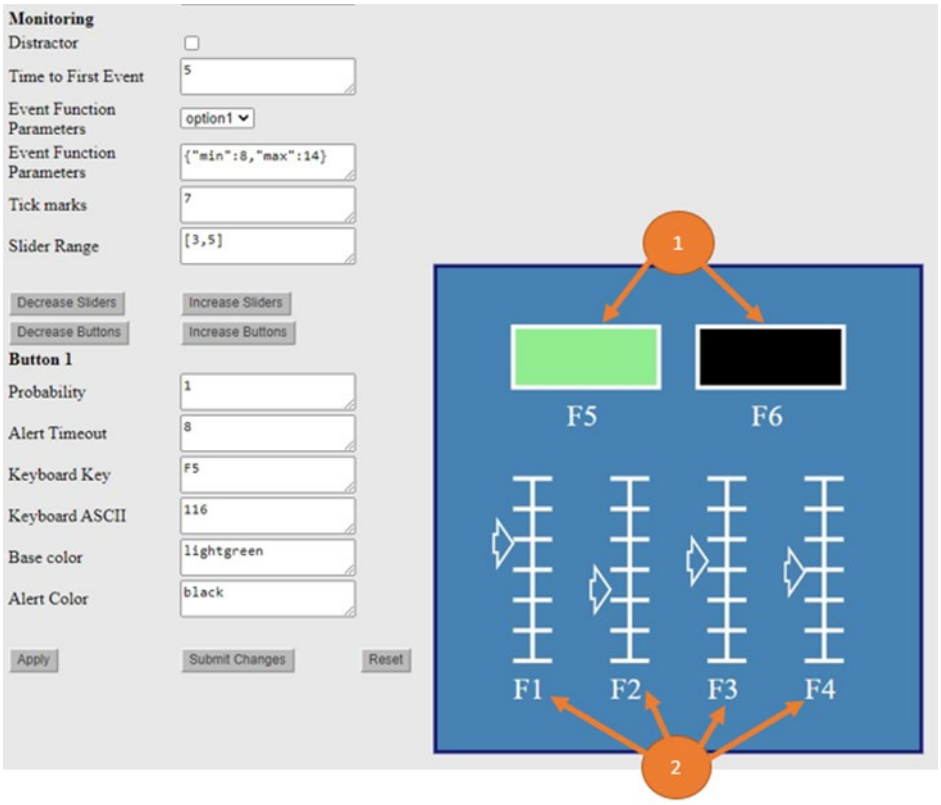


Figure 39. Configuration Options for Monitoring Task

The Monitoring Task consists of two parts:

1) **Buttons:** Each button has a base and alert color. The buttons' base colors are visible at the start of the task but will temporarily change color to indicate an alert. When in an alert status, participants can reset the button by either clicking on it or pressing the key on a keyboard associated with the button. The associated key is indicated below the button, F5 and F6 in the Figure 39 example.

2) **Sliders:** Below the buttons are four vertical lines with tick marks and an arrow that moves up and down the line. Ordinarily, the slider moves within a normal range but during an alert the slider moves outside of this range. When this happens, the participant can reset the slider by clicking on it or by pressing the keyboard key associated with the slider. The associated key is marked below the slider (e.g., Figure 39, keys F1 – F4).

Researchers have a variety of configuration options available for this task:

- **Distractor:** Checking this box turns the task into a distractor, meaning that the color of the buttons will not change and targets on the vertical ranges will never exit the appropriate region. As a distractor, this task will never require input from the participant and will simply serve as a visual stimulus.
- **Time to First Event:** This is the number of seconds between the start of the task and the first alert.
- **Event Function:** This setting is used to control how alerts will be presented over the course of the task.
    - Option 1 – Alerts will be presented at random intervals, within the established alert and event function parameters min and max.
    - Option 2 – Alerts will be presented at an interval set by the researcher using the geometric distribution of alerts with an average time set by the event function parameter *avg_wait*.
- **Event Function Parameters:** This box contains the parameters used by the event function. If the random function (option 1) is selected *min* and *max* times should be entered. If geometric distribution is selected (option 2) then *avg_wait* should be entered.
- **Tick Marks:** This is the number of tick marks on each slider.
- **Slider Range:** This is the base range for the sliders when they are not in an alert state. For the example in Figure 39, participants should be told to respond when the target either moves above the third mark or below the fifth.
- **Decrease Sliders** / **Increase Sliders:** These buttons can be used to modify the number of vertical sliders included in the task for this condition.
- **Decrease Buttons** / **Decrease Buttons:** These buttons can be used to modify the number of buttons included in the task for this condition.

The experimenter can adjust some settings for each button individually. Settings can be shown by clicking on the button in the configuration page:

- **Probability:** This is the probability that this button will be chosen for an alert. The probabilities of all buttons and sliders included in the task are cumulative. For example, if there are four sliders and two buttons, all with probabilities set at "1", then the probability that any of the six targets will be selected is 1/6 or 16.7%. However, if the probability of one button is changed to "2", then the likelihood of that button being used to represent an alert is 2/7 (28.6%), whereas the likelihood for the other five targets being used decreases to 1/7 (14.3%).
- **Alert Timeout:** This is the amount of time in seconds that the participant has to respond to an alert. If the participant fails to respond, the target will reset automatically.
- **Keyboard Key:** The keyboard key that can be used to enter a response, indicated below the corresponding button.
- **Keyboard ASCII:** The JavaScript event keycode of the key to be associated with the button. Note that this is different from a standard American Standard Code for Information Interchange (ASCII) value.
- **Base Color:** The color of the button when it is not an alert.
- **Alert Color:** The color of the button when it is an alert.

Researchers may also adjust some settings for sliders individually. These can be shown by clicking on the desired slider while on the configuration page:

- **Probability:** This is the probability that this slider will be chosen for an alert. The probabilities of all buttons and sliders included in the task are cumulative. For example, if there are four sliders and two buttons, all with probabilities set at "1", then the probability that any of the six targets will be selected is 1/6 or 16.7%. However, if the probability of one slider is changed to "2", then the likelihood of that slider being used to represent an alert is 2/7 (28.6%), whereas the likelihood for the other five targets being used decreases to 1/7 (14.3%).
- **Slider Interval:** The number of milliseconds for the slider to move one tick.
- **Keyboard Key:** The keyboard key that can be used to enter a response, indicated below the corresponding slider.
- **Keyboard ASCII:** The JavaScript event keycode of the key to be associated with the slider. Note that this is different from a standard ASCII value.

5.3.8 Resource Task

The objective for this task is to keep the main tanks' resource level (Figure 40, item 1) as close to the center of the target region as possible by opening and closing gates.

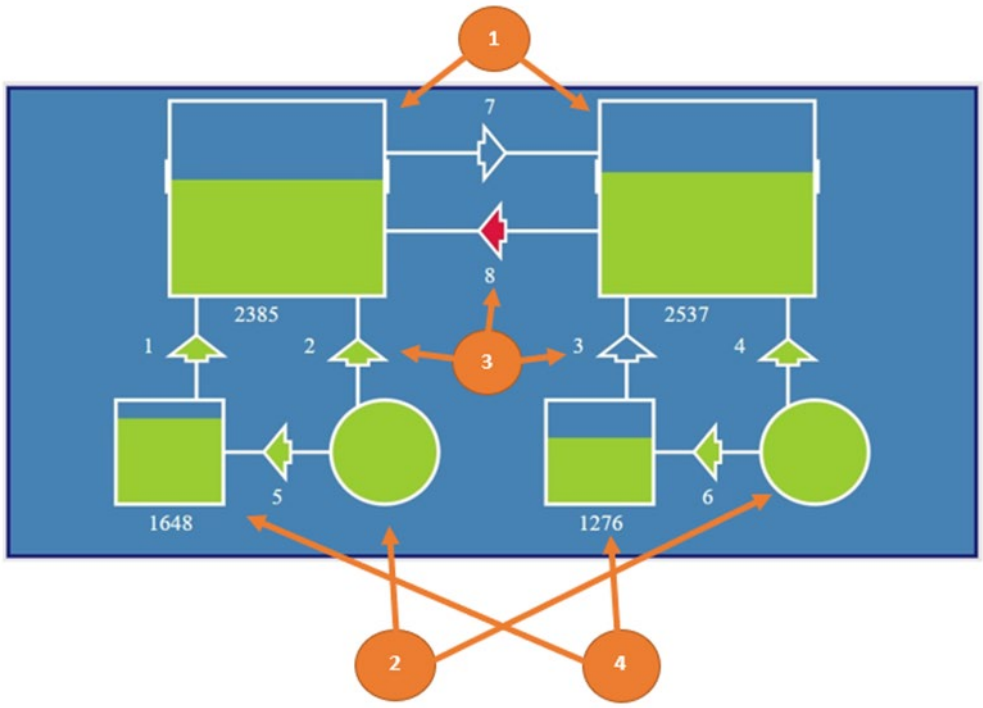


Figure 40. Configuration Options for Resource Task

The Resource task consists of:

1) **Main Tanks:** These tanks have a resource level represented by a green area and a number below the tank. Target range is indicated by the thicker bars on the sides of the tanks.
2) **Fuel Sources:** These circles do not have a set resource level but instead they represent an infinite resource.
3) **Switches:** These operate as gates and can be opened to allow resources to flow in the direction of the arrow. The participant can open a gate by clicking on it or pressing the number next to the switch. A green gate indicates a gate that is currently open, while a hollow gate means it is currently closed. Red gates are broken and cannot be opened by the participant, which is called an alert.
4) **Reserve Tanks:** Resources are not pulled from these tanks unless the corresponding gates are open, allowing the resource to flow out of it. The numbers located below these tanks indicate their current resource level.

There are several settings that the experimenter can adjust on this task (Figure 41).

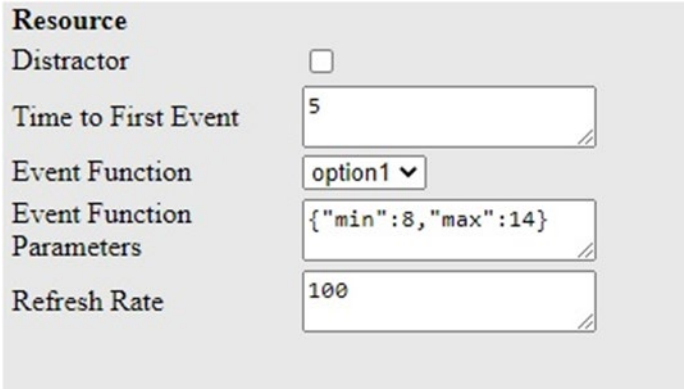


Figure 41. Configuration Options for Resource Task

- **Distractor:** Checking this box turns the task into a distractor, meaning the switches will not break and the main tanks' resource levels will not decrease below the target range. No alerts will be generated, so the participant will never have to interact with the task.
- **Time to First Event:** The number of seconds from the start of the task until the first alert.
- **Event Function:** This setting is used to control how alerts will be presented over the course of the task.
    - Option 1 – Alerts will be presented at random intervals, within the established alert and event function parameters min and max.
    - Option 2 – Alerts will be presented at an interval set by the researcher using the geometric distribution of alerts with an average time set by the event function parameter *avg_wait*.
- **Event Function Parameters:** Parameters used to modify the event function. If the random function (option 1) is selected, the min and max parameters must be specified to control the number of seconds from the start of one alert to the beginning of the next. If the geometric function is selected (option 2) then the desired average amount of time (i.e., *avg_wait*) must be specified as this controls the average number of seconds from the start of an alert to the start of the next one.
- **Refresh Rate:** Setting used to control how often the task refreshes. Each time the page refreshes, three things will happen:
    1) The task state is recorded.
    2) The main tanks decay an amount determined by their settings and refresh rate.
    3) The open switches transfer an amount of resource based on their settings.

There are also several settings that can be adjusted for each tank individually (Figure 42A). These settings can be adjusted by clicking on the outline of the tank or the green resource area while configuring the task.

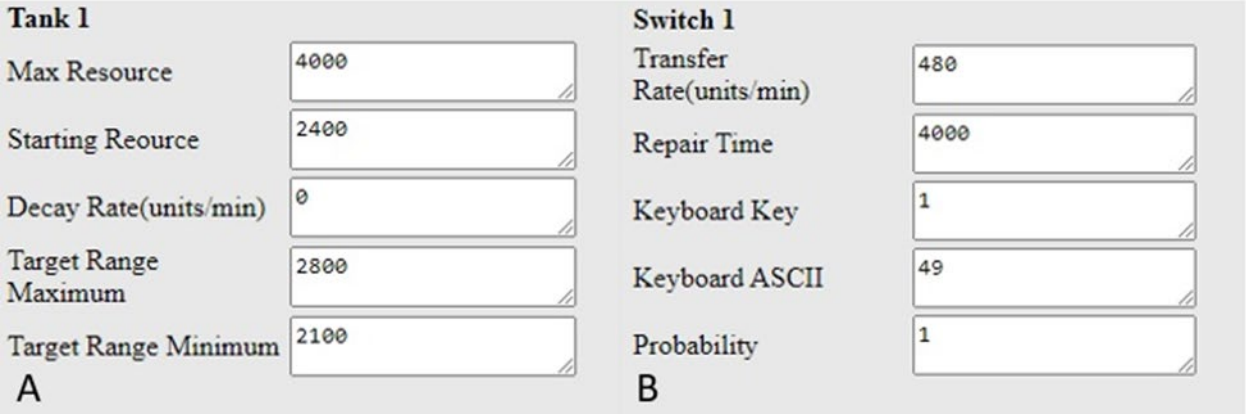


Figure 42. Configuration Options for Tanks (A) and Switches (B) in the Resource Task

- **Max Resource:** The number of units in the tank when it is full.
- **Starting Resource:** The number of units in the tank when the task starts.
- **Decay Rate (units/min):** The number of units that the tank decays in a minute. By default, this value is 0 for the reserve tanks, but it can be changed.
- **Target Range Minimum and Target Range Maximum:** Sets the target range of the task, as indicated by the bold lines on the sides of the tanks. These settings are not available for any tanks.

There are several settings that can be adjusted for each switch individually (Figure 42, B). These settings can be adjusted by clicking on the desired switch while on the configuration page:

- **Transfer Rate (units/min):** The number of units the switch will transfer in a minute when it is on.
- **Repair Time:** The number of milliseconds before a switch will reset after changing for an alert. This is similar to the "Alert Timeout" used in other tasks.
- **Keyboard Key:** The keyboard key corresponding to that specific switch.
- **Keyboard ASCII:** On the switch, the ASCII value of the key that needs to be pressed to control the opening and the closing of the gate.
- **Probability:** This is the probability that this switch will be chosen for an alert. The probabilities of all the switches are cumulative. For example, if "1" is entered for each of the eight switches, then the probability that any of them will be selected is 1/8 or 12.5%. However, if the probability of one switch is changed to "3", then its likelihood of being selected changes to 3/10 (30%), while the likelihood of all other switches decreases to 1/10 (10%).

5.3.9 Tracking Task

The Tracking Task consists of a series of paths (Figure 43, label 1), each with a satellite (label 2) moving along it.

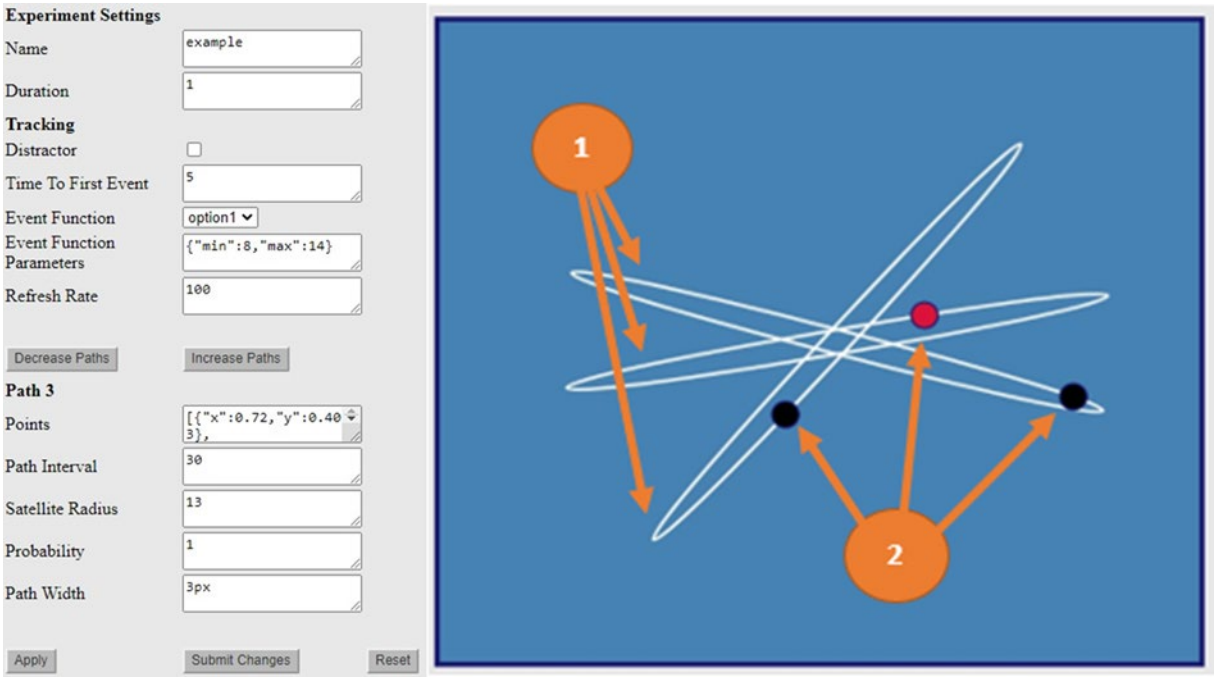


Figure 43. Tracking Task Configuration Options

A satellite will randomly turn red, indicating an alert. When this happens, participants should then click on the satellite, turning it green, then track it by keeping their mouse over the satellite as it moves. Participants should continue to track the satellite until it or another satellite turns red, then repeat the process. Using a keyboard to enter responses to other concurrent tasks will enable participants to continue tracking the satellites with a mouse.

The configuration settings for the Tracking task are similar to other ModME tasks:

- **Distractor:** Checking this box turns the task into a distractor, meaning the satellites will not change to red. With this setting, participants never need to interact with the task so it will serve as a basic visual stimulus.
- **Time To First Event:** The number of seconds from the start of the task until the first alert.
- **Event Function:** This setting is used to control how alerts will be presented over the course of the task.
    - Option 1 – Alerts will be presented at random intervals, within the established alert and event function parameters *min* and *max*.
    - Option 2 – Alerts will be presented at an interval set by the researcher using the geometric distribution of alerts with an average time set by the event function parameter *avg_wait*.
- **Event Function Parameters:** Parameters used to modify the event function. If the random function (option 1) is selected, the min and max parameters must be specified to control the number of seconds from the start of one alert to the beginning of the next. If the geometric function is selected (option 2) then the desired average amount of time (i.e., *avg_wait*) must be specified as this controls the average number of seconds from the start of an alert to the start of the next one.
- **Refresh Rate:** Sets the number of milliseconds between each system state recording.
- **Decrease Paths / Increase Paths:** These buttons modify the number of paths by 1. Paths are added to the top left corner of the task box with a satellite. Note that each additional path also adds a corresponding satellite. New paths are added to the same location [{"x":0.2,"y":0.8},{"x":0.2,"y":0.9},{"x":0.1,"y":0.9},{"x":0.1,"y":0.8}], and so will overlap until assigned to another location.

There are some settings that are configured for each path and satellite set independently. These can be set by clicking on the path or satellite combination while configuring the task:

- **Points:** This field contains an array of points that determine the location of a given path. Each point has an x and y property, and these represent the position of the point, such that (0,0) is the lower left corner and (1,1) is the upper right corner. Each path is comprised of four points, such as the following paths included in the task:
    - Path 1: {"x":0.658,"y":0.821},{"x":0.357,"y":0.152},{"x":0.342,"y":0.179},{"x":0.643,"y":0.848}
    - Path 2: {"x":0.277,"y":0.442},{"x":0.720,"y":0.598},{"x":0.723,"y":0.558},{"x":0.280,"y":0.402}
    - Path 3: {"x":0.720,"y":0.403},{"x":0.715,"y":0.364},{"x":0.280,"y":0.597},{"x":0.285,"y":0.636}
- **Path Interval:** A measure of speed, this is the amount of time (in seconds) necessary for a satellite to complete one trip along all four points in the path.
- **Satellite Radius:** Used to configure the size of the satellite. The default is 13.
- **Probability:** Probability that the selected satellite will be chosen for an alert. The probabilities of all objects in the task are cumulative. For example, if there are four satellites and each is assigned a probability of "1", then the probability of any path being selected is 1/4 or 25%. However, if the probability of one satellite is changed to "3", then the probability of the selected satellite becomes 3/6 (50%), and the probabilities of the other satellites decrease to 1/6 (16.7%).
- **Path Width:** Controls the thickness of the line used to represent the selected path. Note that a value of "0px" will make the path disappear.

5.3.10 Aircraft Coordination Task

As shown in Figure 44, there are a variety of configuration options possible with the Aircraft Coordination Task. Foremost are the time until the second aircraft enters the screen, the time between aircraft, speed of the center aircraft, and desired means for entering a response.

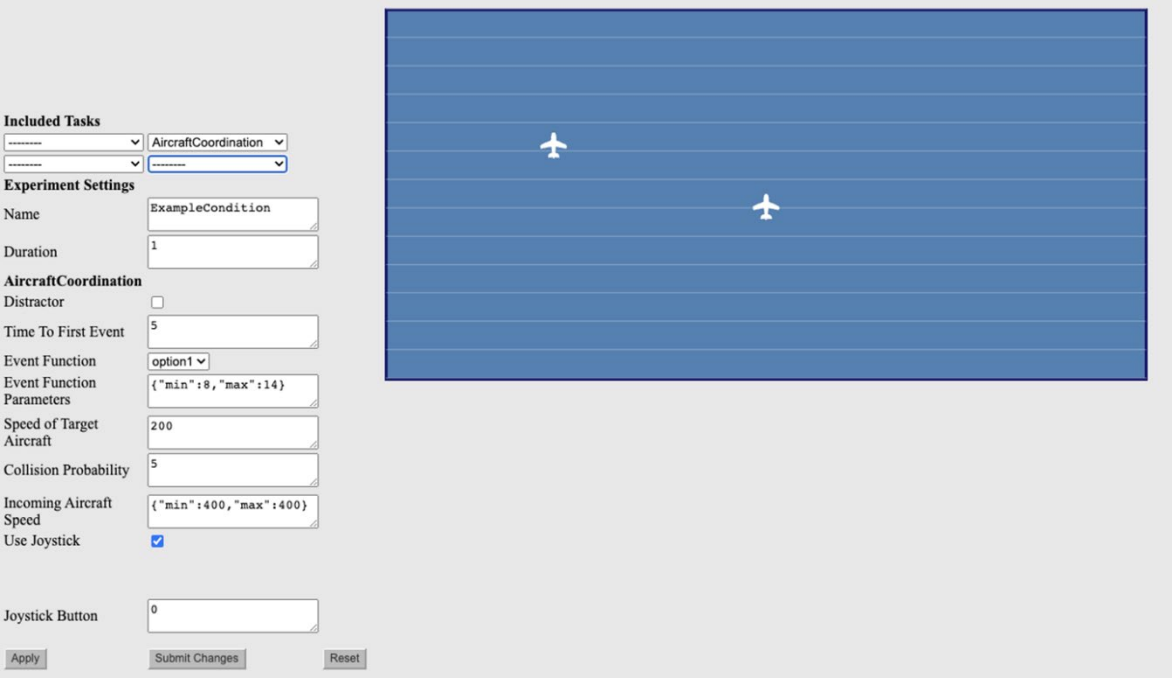


Figure 44. Configuration Options for the Aircraft Coordination Task

- **Distractor:** Checking this box turns the task into a distractor, meaning the satellites will not change to red. With this setting, participants never need to interact with the task so it will serve as a basic visual stimulus.
- **Time To First Event:** The number of seconds from the start of the task until the first alert.
- **Event Function:** This setting is used to control how alerts will be presented over the course of the task.
    - Option 1: Uniform distribution
        - Syntax: {“min”:8,”max”:14}
        - Min: Denotes the minimum amount of time, in milliseconds, from the start of one alert event to the start of the next
        - Max: Denotes the maximum amount of time, in milliseconds, between the start of one alert event to the start of the next
    - Option 2: Geometric distribution
        - Syntax: {“avg”:8}
        - Avg: Denotes the average amount of time, in milliseconds, from the start of an alert event to the start of the next alert event

There are also a variety of object settings which can be used to control aircraft speed.

- **Speed of Target Aircraft:** The speed at which the focal aircraft in the center of the display appears to move. In actuality, this configuration option adjusts the speed of lines in the background.
- **Collision Probability:** Adjusts the odds of a collision event occurring between the two aircraft. In other words, “100” would mean that the aircraft would collide on each trial, “50” would mean they collide on half the trials, etc.
- **Incoming Aircraft Speed (min, max):** Adjusts the speed of incoming aircraft. Faster aircraft speed will require faster responses from the participant to avoid a miss.
- **Use Joystick:** A checkbox used control whether participants will respond using a keyboard (unchecked) or will be responding through a Universal Serial Bus (USB) controller or joystick (“1”) or not (“0”). When this option is unchecked, another field will appear below the checkbox asking researchers to enter the ASCII value of the keyboard key to which they want to bind participant responses.

5.4 Selecting the Number of Tasks Displayed

Conditions can also be configured with regard to the number and placement of tasks displayed concurrently. The traditional quad task display is presented in Figure 33, but other layouts such as the tri-task display (Figure 45, A), dual task perpendicular (B), and dual task diagonal (C) may also be used.

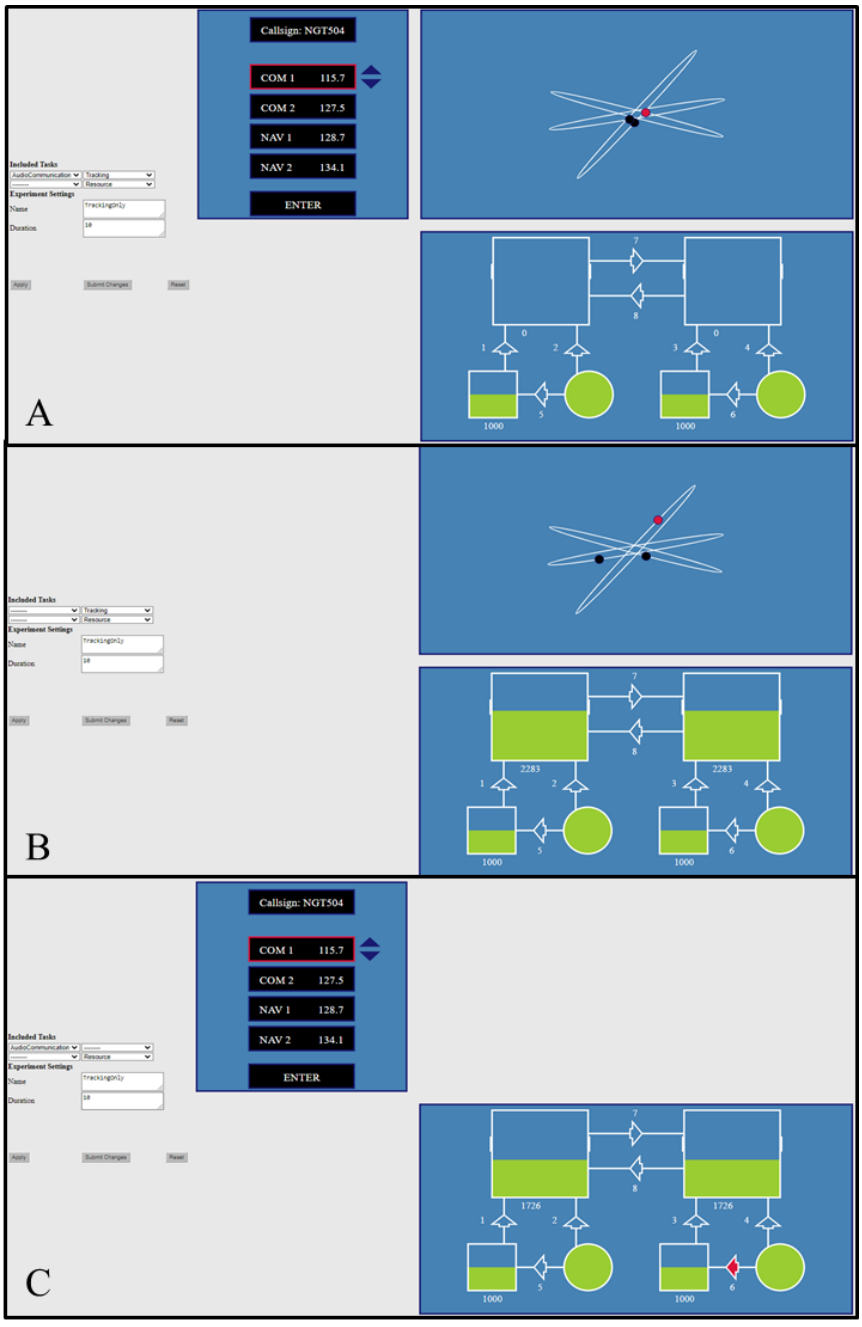


Figure 45. Three Possible Task Layouts for Use in a Condition

As noted, if fewer than four tasks are needed for a condition, the "Place Holder" option may be selected to maintain the same task dimensions as when four tasks are present. In the example of Figure 45B, this would mean that the Tracking and Resource tasks would be shown to participants at the same size as they would see the condition in Figure 33. Alternatively, if the drop-down options for both the first and third quadrants were changed to being undefined (i.e., ---------), the Tracking and Resource tasks as shown in B would automatically be resized to fill the center of the screen.

6.0 EXPERIMENT ADMINISTRATION OVERVIEW

6.1 Experiment Administration Page

This is the main Experiment Administration Page to set up the organization and instructions for ModME experiments. There are 10 main components to this page (Figure 46):

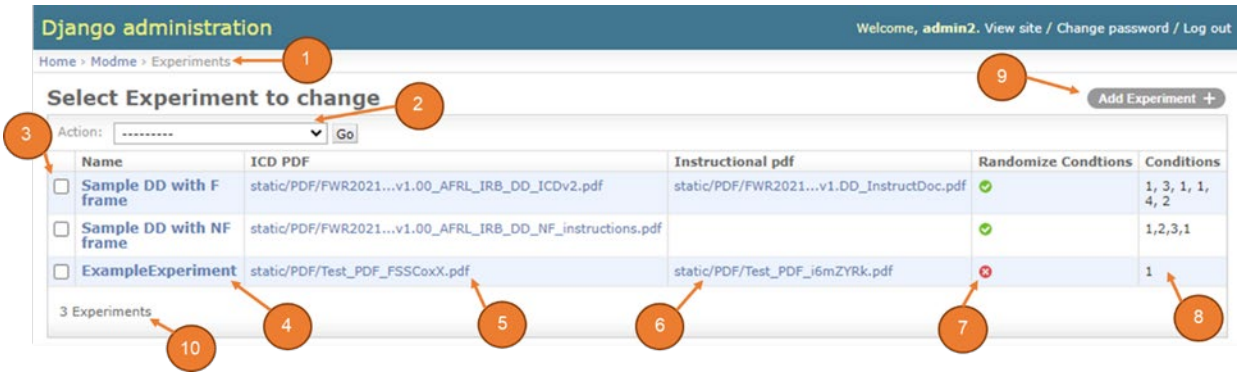


Figure 46. Main Components of the Experiment Administration Page

1) **Location**: Indicates current page while logged into the Administration Site. In the screen shot above, the current location is *Home > Modme > Experiment*.

2) **Action Selection**: Drop-down field to select "actions" that may be taken on the experiments listed (Figure 47). At the time of publication, the only action available is to delete items. To do this (or any other action as they become available) select an experiment in the Action Item Selection Column, select the desired action, and then press "Go." A confirmation pop-up will appear on the screen asking for confirmation, at which point the requested action will be completed.

Figure 47. Zoomed View of Action Selection

3) **Action Item Selection Column**: These are the check boxes in the left column of this table. Multiple experiments will need to be selected in this column before accessing the Action Selection tool.

4) **Name Column**: Lists the name assigned to each experiment.

5) **ICD PDF Column**: The full file path and name of the informed consent document (ICD) linked to each experiment.

6) **Instructional PDF Column**: This is the full file path and name of the instructional document linked to the experiment. Note that these are not instructions for completing the individual tasks but are instructions specific to each experiment used to explain each step in greater detail to participants.

7) **Random Column**: This defines whether the conditions are presented in a random order. As used in ModME, conditions are the individual task sessions that comprise the larger experimental session.

8) **Conditions Column**: Lists conditions included in an experiment. When the Random indicator for an experiment is "false", the ordering of each condition in the experiment will also be shown.

9) **Add Experiment**: Click here to add an experiment.

10) **Instance Counter**: This tracks the number of experiments in this ModME account.

6.2 Creating a New Experiment

To create a new experiment, click on the "Add Experiment" button, which will then open the form shown in Figure 48. This form will walk through each step of creating a new experiment. This same page can also be accessed from the list of tables by clicking on the "Add" button to the right of the Experiments link.

Figure 48. Form Components Available when Adding an Experiment to ModME

1) **Name**: This name will not be visible to participants, and it should be unique and descriptive. Any alphanumeric characters can be used here; however, it is strongly advised to make this meaningful, descriptive, and short. It is also advised to include an indicator of date (e.g., "June 2020"). Additionally, if there are multiple versions of the same experiment (e.g., both remote and in-lab), a good practice is to include some unique identifier for each variation.

2) **ICD PDF:** The ICD approved by an Institutional Research Board (IRB) or other similar governing body should be uploaded here, and it can only be uploaded as a PDF file. The ICD will be displayed to participants prior to the Instructional Page or start of the experiment. If participants do not need to indicate consent for this part of the study (e.g., the task is part of a larger battery and informed consent has already been given), this page can be skipped by selecting the "Skip ICD Page" option below. Customize the text for the participant ICD confirmation in the "ICD Confirmation Text" below.

    a) To upload an ICD (it must be a PDF), select the "Choose File" button. A file window will appear. Find and select the corresponding document, then select "Open." This action will save an instance of the document within ModME. This document will start with the same title as the original file but with a unique, system-generated code appended to the end. This code will help track changes to this document field to aid version control.

    b) To select a different document, return to this experiment and look for the "Change" label, then click on the "Choose File" button beside it, then navigate to and select the appropriate file using the same process as before.

    c) To remove the document without replacing it, go back to the experiment and, under the field, select a check box next to the word "Clear", then navigate to the bottom of the page and click on the "Save" button.

3) **Skip ICD Page**: Selecting this will skip the ICD page. As noted elsewhere, this should ONLY be done if participants are provided a physical copy of the ICD and are properly consented in another way.

4) **ICD Confirmation Text**: This will be presented below the ICD and is included as an opportunity to ask participants to confirm they read the ICD, all questions have been answered, and they wish to continue with the experiment. A sample question could be:

    > *I have read the above information, affirm that I meet all eligibility criteria* [repeat inclusion criteria here]*, and agree to participate in this study.*

5) **Instructional PDF**: This page will be displayed after the ICD, unless the option to not include an ICD has been selected, or if the option to "Skip Instruction page" has been selected. Any instructions specific to the experiment should be uploaded here, and as with the ICD, this file must be saved as a PDF. Additionally, participants may be required to confirm they understand the instructions before proceeding to the experiment. If included, this question will be presented to participants on the same page, immediately below the instructions. Text can be customized using the "Instructional Confirmation Text" below.

    a) To upload instructions, click on "Choose File", select the file to be added, then click "Open," Once uploaded to ModME, this saved document will start with the same title as the original file, but a unique code will be appended to the end which enables researchers to track changes and thereby aid version control.

    b) To select a different document, click on the "Choose File" button located next to the "Change" label, then repeat the process used to select the original document.

    c) To remove the document without replacing it, select a check box next to the word "Clear", then navigate to the bottom of the page and click on the "Save" button.

6) **Skip Instruction Page:** Selecting this will skip the instruction page. This should only be used when participants have already been trained on the task or received instructions in another way.

7) **Instructional Confirmation Text:** Appears below the Instructional Documentation and is used for messages such as "Once you have read the instructions above, click on the 'Next' button to continue with the experimental session."

8) **Conditions:** Enter the identification (ID) number of each Condition in the desired order (e.g., 1,3,1,2). To find the ID number of a condition, navigate to the Condition Admin page [Home > Administration > Condition], then identify the number corresponding to each of the desired conditions (Figure 49).

9) **Randomize Conditions:** Using this function will allow for the order of the conditions to be presented to participants in a random order, rather than either a specified order or the order in which the conditions were created.

10) **Experiment Ending Message**: This text will be presented to participants at the end of the experimental session. Content here could include thanking participants for their time, providing a short debrief statement, letting them know how or where to submit the completion code, or giving them instructions for the next session. For example:

> *Thank you for participating in our experiment.*
>
> *Copy your individual completion code, located below, and then close the browser.*
>
> *To claim your completion code, go to www.ExampleWebsite.com and enter your individual completion code.*

11) **Experiment Notes**: This is an optional field included as an opportunity to attach notes specific to the experiment, such as studies or date ranges for which it was used. Content here will not be visible to participants, and the only restriction is that all text must be alphanumeric.

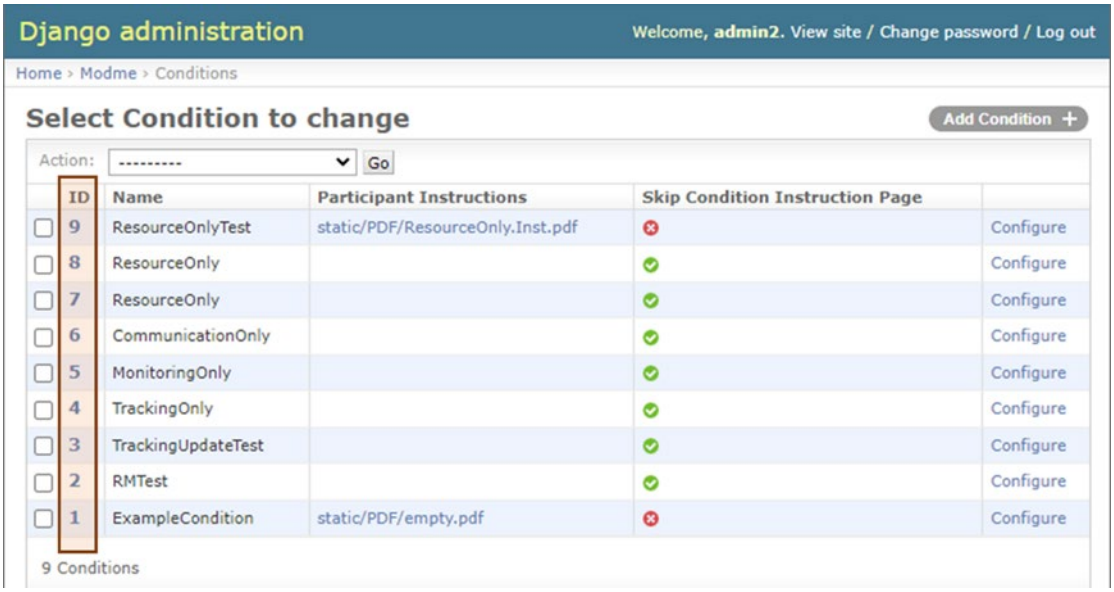


Figure 49. Sample List of Available Conditions

7.0 EXPERIMENT LOOP OVERVIEW

7.1 Experiment Loop

This section of the guide explains all elements participants will experience in the testing environment. The "Experiment Loop" refers to the order in which participants move through the experiment and is important to understand when building and running experiments. Figure 50 outlines the Experiment Loop process used in ModME.

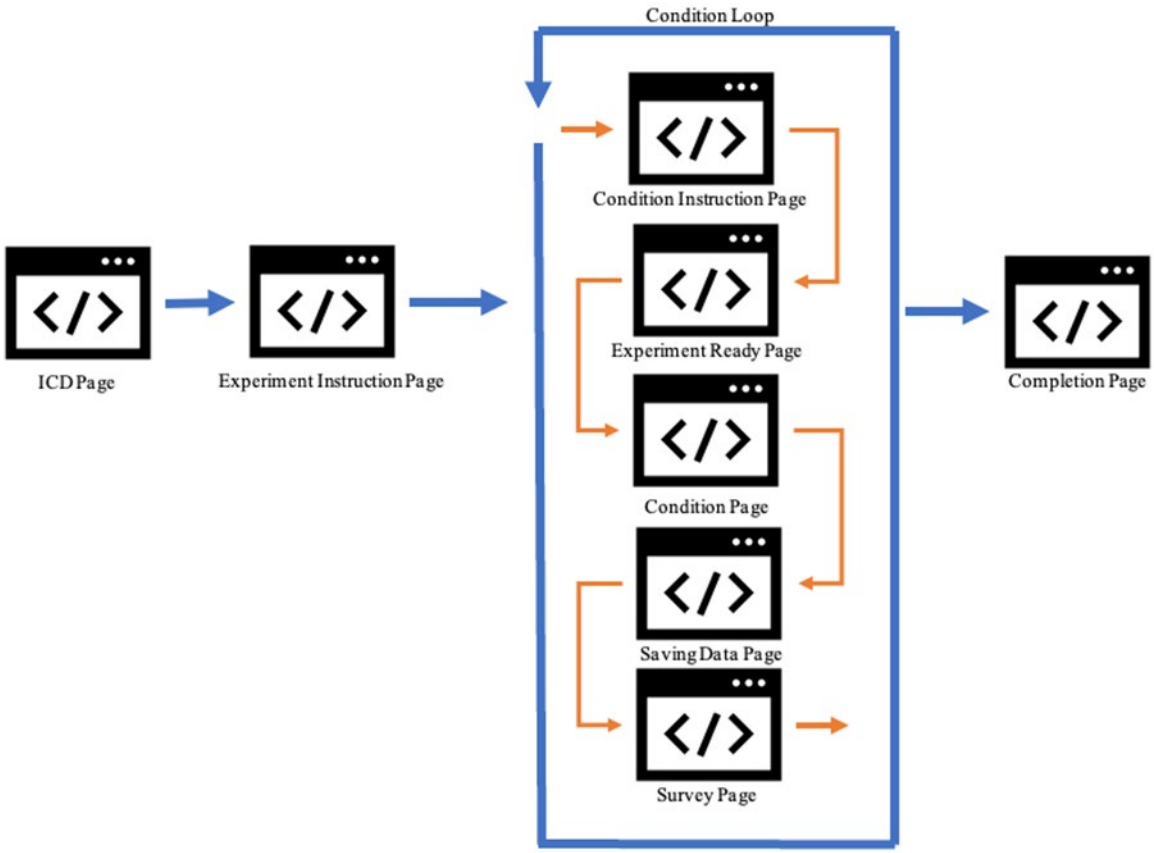


Figure 50. Experiment Loop Process

7.1.1 Informed Consent Document Page

The ICD page will usually be the first page participants encounter if they have not already consented to participate in the study through other means. This page is intended for the participant to read a general description of the study, expectations about their performance, and their rights as a human subject. It is possible to skip this page for an experiment by selecting "Skip ICD Page" in the Experiment Form. This page may appear different depending on the browser used. Participants have multiple options for viewing the ICD, comparable to those for most PDF files read online, and they may download or print the document if desired. At the bottom of the webpage will be ICD Information Confirmation followed by a button labeled "Next".

7.1.2 Experiment Instructions Page

If included, participants will see the "Experiment Instructions Page" immediately after the ICD. This file is an opportunity for researchers to offer greater detail about the study than is appropriate for the ICD, such as what participants will encounter during the task and screen shots to improve their familiarity before starting. This page may appear different and have different options depending on the browser used.

7.2 Condition Loop

The next phase is called the "Condition Loop", and it is the only aspect of the larger Experimental Loop that is not necessarily linear and can be modifiable to fit experimental needs. Within the Condition Loop, each condition cycle executes consecutively and automatically.

7.2.1 Condition Instruction Page

Each condition included in a given experiment starts with the Condition Instruction Page. Conditions included in an experiment represent different task settings or configurations (e.g., time allowed, arrangement of tasks, etc.), so this page would be used to explain any differences between conditions to participants. Alternatively, if the experimental session is quite long, this page may be used to allow participants a short break. Once they have finished reading the document, participants should click the "Next" button at the bottom of the screen to continue with the session. If this page is not relevant or is unnecessary, it can be excluded by using the "Skip Condition Instruction Page" option.

7.2.2 Condition Ready Page

After participants have reviewed the "Condition Instruction Page" (if included), they will continue to the Condition Ready Page. This page exists simply as a pause before continuing on to the experimental task. Note that this page is hard-coded into the program and cannot be skipped.

7.2.3 Condition Task Page

After the Condition Ready Page, participants will proceed to the Condition Task Page to work on the condition. The inclusion and arrangement of tasks can be modified as needed. For an explanation of how to tailor conditions as needed, please refer to the Condition Administration section (5.0) of this document.

7.2.4 Data Saving Page

The Data Saving Page (Figure 51) will appear immediately after completing each condition within an experiment, and the step is necessary to properly transmit all data gathered during the session. In most cases, this process will take no longer than 5 seconds, but depending on the condition completed it is possible for this screen to flash so quickly that participants may not see it.

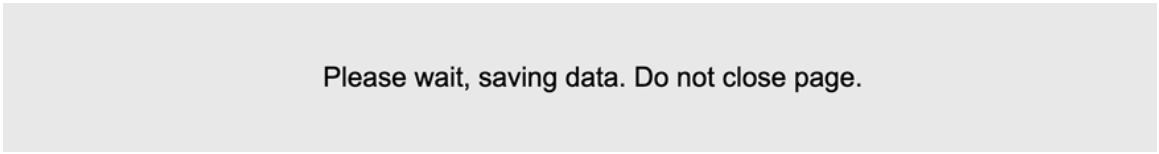


Figure 51. Sample Notification for the Data Saving Page

How long the process of transferring data from a condition takes depends on several factors, primarily:

- Duration of the experiment task
- The type and number of conditions included in the condition
- Speed of the participant's internet connection

If there are concerns regarding the amount of time necessary to save data to the database or the quality of participants' internet connection, the best resolution is to limit the duration of conditions. For this reason, it is recommended that conditions be limited to 15 to 20 minutes in duration, as longer sessions may contribute to issues with data storage. Additionally, participants may become impatient if the program lingers too long on this page, incorrectly assuming there is a problem with the task and ultimately closing the page.

To modify the condition duration, navigate to the Administration Page, then go to Conditions [Admin > Conditions]. From the list shown, select the ID number of the condition to be changed, and then the Change Form will appear. Any desired modifications can then be made to the "Length of experiment in minutes" field. For more detailed information about creating and modifying conditions, please refer to the Condition Administration section (5.0).

Once all data have been saved to the database, the page will automatically progress to the National Aeronautics and Space Administration Task Load Index (NASA-TLX) survey.

7.2.5 Survey Page

The NASA-TLX survey page asks participants to report their opinion regarding the difficulty of the condition just completed. Participants should indicate their responses using the scale located below each survey item, ranging from "Very Low" to "Very High" or "Perfect" to "Failure", then click on the "Submit" button when they are done (Figure 52).

If the experimental session includes multiple conditions, participants will automatically continue to the next condition until all have been completed. After the final condition, the session will exit the "Condition Loop" and participants will be directed to the Completion Page.

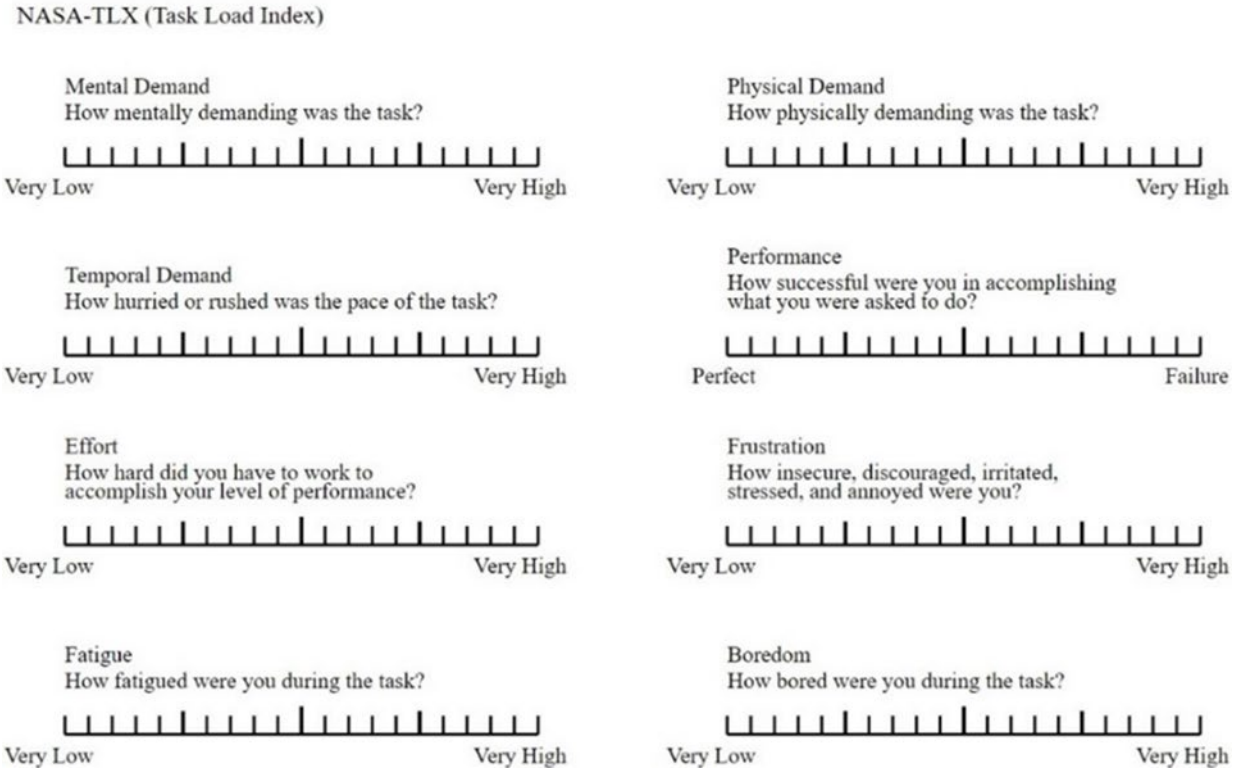


Figure 52. NASA-TLX as Shown in ModME

7.2.6 Completion Page

This is the final page in the experiment and should be used to thank participants for their time and, if appropriate, provide a code they can use to confirm they completed the experimental session (Figure 53). The inclusion and content of these messages can be modified, depending on the design of any given study. For example, compensation codes are not necessary when the task is performed in the laboratory, whereas participants will not need to "alert an experimenter" if they complete the session at home.

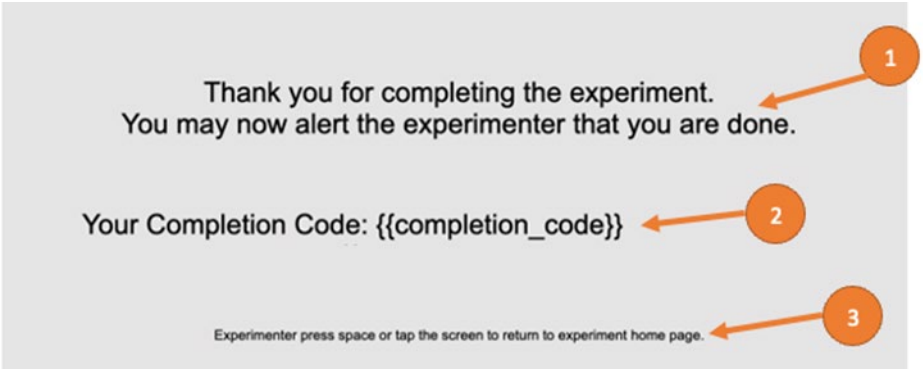


Figure 53. Screen Shown to Participants After Completing an Experiment

1) **Experiment Ending Message:** This is the "Thank You" message, confirming to participants that they have completed the session.

2) **Completion Code:** This functionality is included for remote or off-site data collection as a means for participants to demonstrate they completed the task and, if appropriate, eligible for compensation. Instructions on this page and elsewhere in the session should remind participants to save this code as it is the only means of confirming they completed the task. Without the code, there is no direct way of linking participants with their data and verifying their performance on or completion of the experimental session.

   In some instances, however, it may be possible for researchers to manually validate participants' completion of the session. The first method is to compare the list of completion codes that have been claimed through the participant management system being used (e.g., Amazon's Mechanical Turk or MTurk) against the list of codes "issued" in ModME. If there is only one unclaimed code, it may belong to the participant in question. In the case of multiple unclaimed compensation codes, data may also be matched with the date or time of data participants completed the task.

3) **Return Screen Message**: This line is included to instruct participants on how to return to the main screen.

8.0 REMOTE EXPERIMENT OVERVIEW

Remote experiments are studies conducted entirely online and outside of a laboratory setting. In most respects, options for these studies are no different than those described in the previous section covering Experiment Loop Overview (7.0). The key difference with remote experiments however is that multiple participant entries must typically be created in advance. This way, a common link can be shared with the sample population (e.g., Prolific, MTurk, etc.), then each participant in the study fills one of the entries created.

8.1 Creating a Remote Experiment

To create a remote experiment, navigate from the Home > Modme page, then click on the link for "Remote Experiments". Once on this page, click on the grey button labeled "Add Remote Experiment +" located on the lower left side of the screen. The next page to load will be the Remote Experiment form (Figure 54).

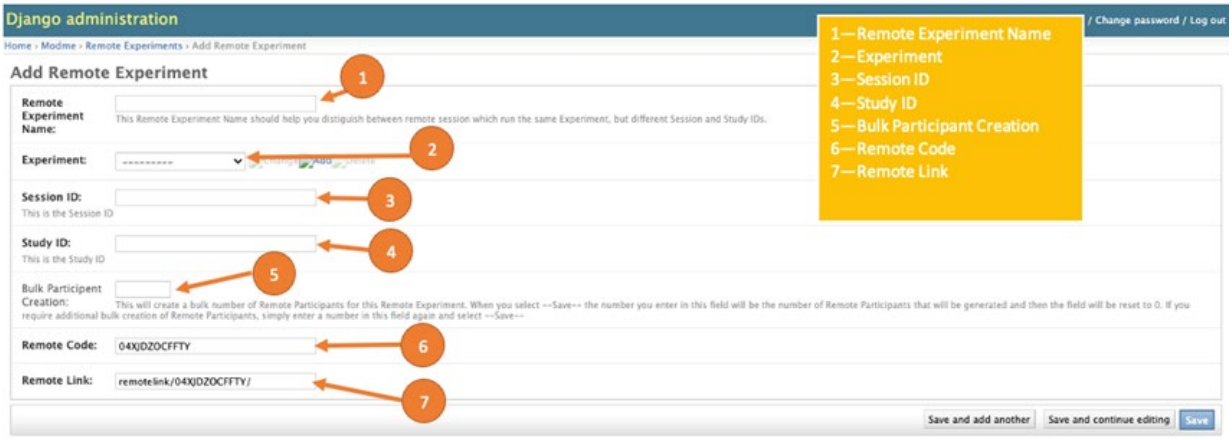


Figure 54. Key Details to be Completed when Creating a Remote Experiment

1. **Remote Experiment Name**: This name will not be visible to participants, and it should be unique and descriptive. Any alphanumeric characters here; however, it is strongly advised to make the name meaningful, distinctive, and short. It is also advised to include a data element, like "June 2020".

2. **Experiment:** Select the experiment to be run remotely. The same experiment may be used in multiple remote studies but note that the base experiment must be created first.

3. **Session ID**: In conjunction with the Study ID, this field is used to distinguish between different Remote Experiments and maintain data organization. Any alphanumeric characters may be entered here but note that the field may not be left blank.

4. **Study ID:** In conjunction with the Session ID, this field is used to distinguish between different Remote Experiments and maintain data organization. Any alphanumeric characters may be entered here but the field may not be left blank.

5. **Bulk Participant Creation**: This function allows for the simultaneously creation of multiple participant entries and the automatic assignment of participants to available entries. New entries are created by entering the desired number of participants (i.e., the target *n*) into the field, then clicking the "Save" button at the bottom of the page. The field will automatically zero out so additional entries may be created in the future.

   To learn how many Remote Participant slots are available, go to the Remote Experiment Admin Page. The right side of the table displays the number of available participant entries for the desired experiment, as well as the number of on hold and completed entries. A participant entry is in a "hold" state if a participant has been assigned to it but has not completed the experiment. For instructions on how to remove holds in a remote experiment, refer to section 8.2, item #7 regarding the "Hold Status Column."

6. **Remote Code**: This code is automatically generated and is used to ensure participants cannot guess or "hack" completion codes and creates a degree of separation between participants and critical experimental information.

7. **Remote Link**: This is the link which will sent to participants. Since the "root" of ModME service will differ depending on how each instance of the program is implemented, the root will be unique for each remote link. The full link is:

   www.[*root address*]/ModME/remotelink/[*remote code*]

   For example, if the instance of ModME service was being hosted on the website *https://www.ExampleUni.edu*, and the remote code were *K4PFWZUIM*, the full link would be:

   www.ExampleUni.edu/ModME/remotelink/K4PFWZUIM

   This full link is what must be given to participants who are taking part in the study remotely. For studies in which the link is specific to an individual participant, be careful to ensure the entire link with the full remote code is provided.

8.2 Remote Experiment Administration Page

This is the Experiment Administration Page to set up the organization and documentation for remote ModME experiments. There are 11 important components to this page (Figure 55):

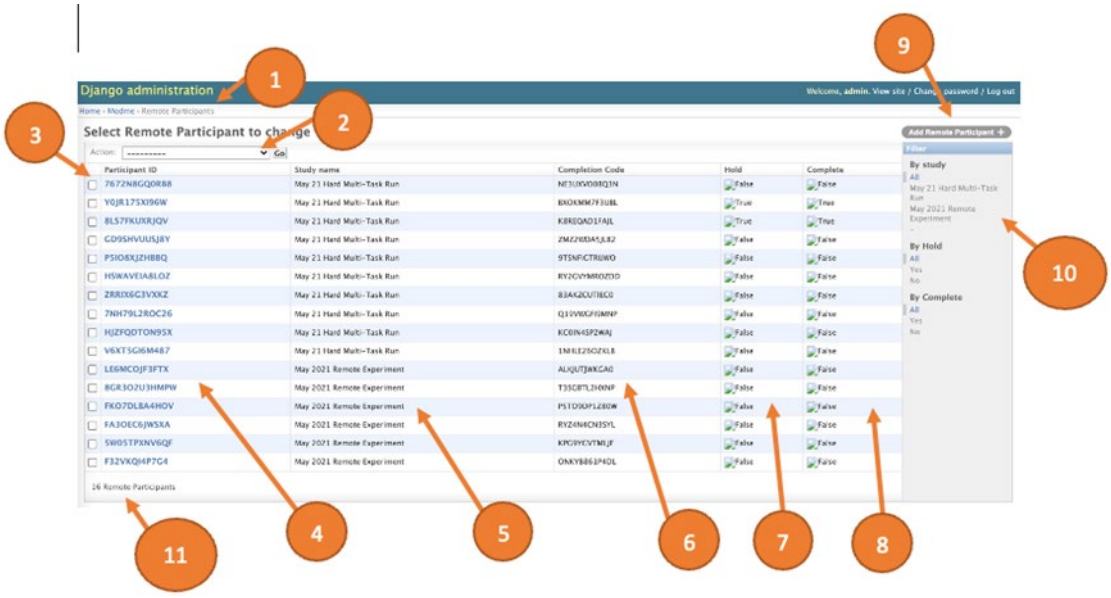


Figure 55. Experiment Administration Page

1) **Location**: This line shows location in the Administration Site. Currently, the location in the image is Home > Modme > Remote Participant

2) **Action Selection**: This is a drop-down field where "actions" may be selected (Figure 56). To perform an action, select an experiment through the Action Item Selection Column, select an action and then press "Go." A confirmation pop-up will appear, and the action will take place once confirmed.

Figure 56. Action Selection Button

3) **Action Item Selection Column**: These are the check boxes in the left-most column of this table. Select one or more of the experiments before selecting an action.

4) **Participant ID Column**: This is the unique, anonymous participant identification code and is not linked to any of personally identifiable information (PII). This is done to ensure confidentiality of data collected.

5) **Study Name Column**: This is the name of the remote experiment to which the participant is linked.

6) **Completion Code Column**: This is the participants' unique completion code and will be displayed on the last page of the experiment. Participants are only given this code if they complete the full experiment. This is only relevant when collecting data using a platform such as Prolific or Amazon's Mechanical Turk.

7) **Hold Status Column**: When a participant uses the remote link and begins the experiment, they are assigned one of the remote participant entries and that entry is set in a hold status. If the field is marked "False," then the participant entry is available to be assigned to a remote participant, whereas "True" indicates it is not available to be assigned.

Once a participant has completed the experiment, both the *Hold* and *Completion Status* for the corresponding entry will change to "True". Ideally participants will complete the experiment in a timely manner, but they may abandon the task prematurely or simply forget to complete it.

To remove these incomplete entries, researchers must either delete the entries or reset the holds. Below are the instructions on how to perform these actions and their effects. Choose which process works best for the planned experiment.

a. Deleting the Holds – Select each entry to be removed by clicking on the corresponding Action Item Selection (3) column. Then choose the "delete selected Remote Participants" in the Action Selection (2). Filter the remote participants by using the Remote Participant Filter (10).

   NOTE: That this will only delete the remote participant from the visible table, and incomplete entry will still appear in the data tables for this experiment. Importantly, this method avoids the potential for confusion with "Removing the Holds" by ensuring the same ID is not assigned to multiple participants.

b. Removing the Holds – Select the individual remote participant then go to the "Hold" field and unselect it. Then click on the "Save" on the bottom of the page for this change to take effect. Again, consider filtering remote participants by using the Remote Participant Filter (10; explained further below).

   NOTE: When releasing this remote entry, it is ready to be assigned to the next participant. Consequently, **the data table will list *two entries with the same participant ID***. Ideally, one will be incomplete and the other will be complete (if the second participant completes the experiment).

c. Allowing Holds to Accumulate – Without taking either of the steps above, the number of holds will accumulate in the system.

   Effects: It might be necessary to generate additional remote participant entries, and this can cause the Remote Participant Table to be unnecessarily confusing and crowded. If implementing this strategy, using the Remote Participant Filter (10; explained further below) allows researchers to manage the number of remote participants entries displayed in the table.

8) **Completion Status Column**: This lists the completion status of each remote participant entry. This will be "False" if the participant has not completed the experiment, and the experiment was completed.

9) **Add Remote Participant Button**: Click here to add remote participant entries to the experiment.

10) **Remote Participant Filter**: Researchers can filter by three different criteria (Figure 57) to identify the desired records more effectively.

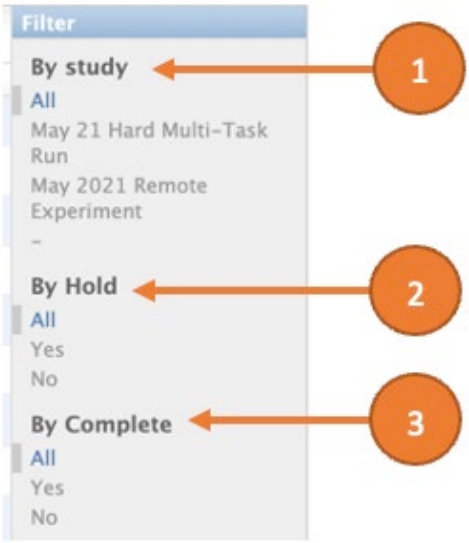


Figure 57. Filter Options for Participant Entries

a. By Study – Filter by Remote Experiment Name. By default, the filter is set to "All" but can be used to list only remote experiments with remote participants. To see only the remote participants from a given Remote Experiment, or "study," select the corresponding experiment name.
b. By Hold – Filter by a participants' hold status. Selecting "Yes" will identify IDs currently on hold and selecting "No" will list the IDs with no hold present.
c. By Complete – Filter by a participants' completion status. Selecting "Yes" will IDs linked to completed experimental sessions, while "No" will return IDs linked to incomplete experimental sessions.

11) **Instance Counter**: Tracks the number of experiments that have been created in this specific instance of ModME.

8.2.1 Creating a New Remote Participant

Entries for remote participants may either be created individually or in bulk. For both, start on the Site Administration page, then click on the link for "Remote Participants" (Figure 58A). An example of the screen where remote participant entries may be added or modified is shown in Figure 58B.

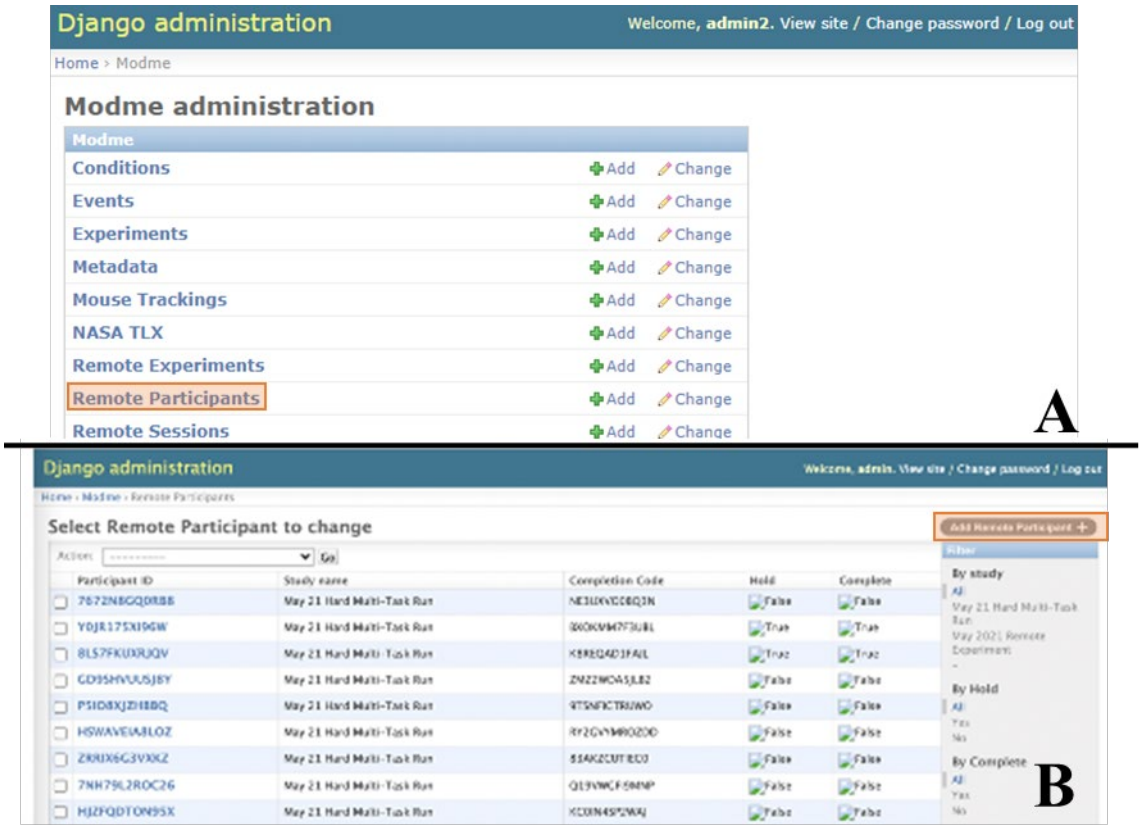


Figure 58. Remote Participants Link on Site Administration Page (A) and Site for Creating/Modifying Participants (B)

8.2.1.1 Creating Entries Individually

To create an individual remote participant, click on the "Add Remote Participant +" button indicated in Figure 58B, which will lead to the form shown in Figure 59:

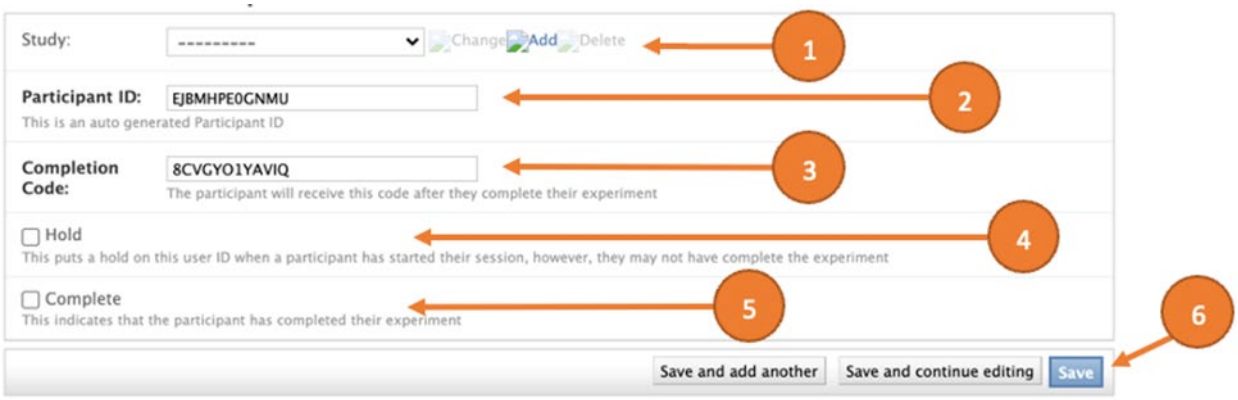


Figure 59. Options Available when Creating a Single Participant Entry

1) **Study**: Select the remote experiment to which this remote participant entry will be linked. The "Change" button allows users to modify the remote experiment selected, whereas the "Add" button will create a new remote experiment form. The "Delete" button will clear the selection if one has made.

2) **Participant ID**: This field is automatically populated with a randomly generated, alphanumeric code unique to each participant. It is strongly advised not to edit this field, though it may be edited when necessary (e.g., adding a prefix or suffix to the ID).

3) **Completion Code**: This field is also automatically populated with a randomly created alphanumeric code unique to each participant. The code will be used when participants are recruited from a service like Mechanical Turk or Prolific to submit as proof they completed the experiment. It is highly advised not to edit this field, though it may be edited when necessary (e.g., adding a prefix or suffix to the completion code).

4) **Hold Status**: By default, this field is unselected and in general should not be changed unless manually releasing the hold due to inactivity or the entry being incomplete.

5) **Completion Status**: This is the completion status of the participant. By default, this field is unselected, and it is strongly recommended to keep this field unselected when creating a new remote participant. This field should not be changed manually unless there is an error in the system that affects the setting of the completion status.

6) **Save Buttons**: This saves the entered data and adds the participant to the list of remote participants or saves the modifications made, as appropriate. This button must be clicked to implement any changes made on the page.

8.2.1.2 Creating Spots in Bulk

To create multiple participant entries at one time, start on the Remote Experiment Admin page (i.e., Home > Modme > Remote Experiment), then select the name of the appropriate Remote Experiment (Figure 60).

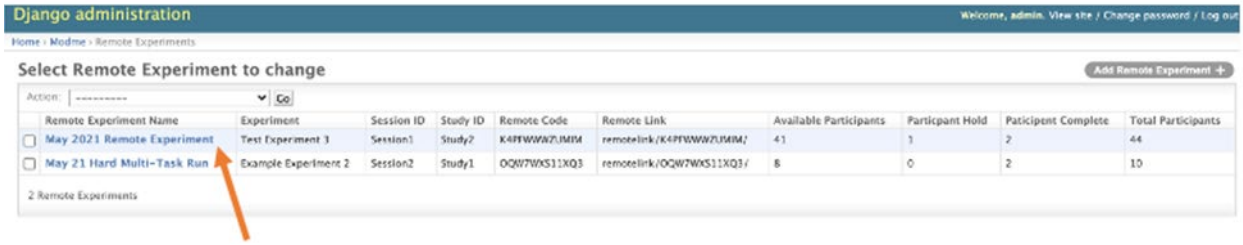


Figure 60. First Step in Bulk Participant Entry Creation

This will bring up the remote experiment entry specific to the desired study (Figure 61). On the next screen, enter the appropriate number of participants in the Bulk Participant Creation field then click the “Save” button.

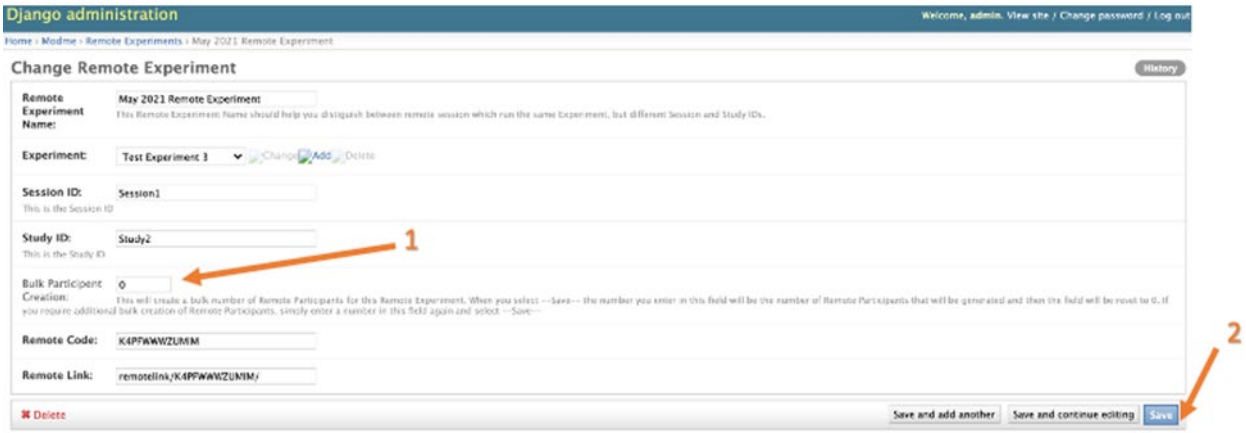


Figure 61. Second Step in Bulk Participant Entry Creation

After selecting “Save,” the Bulk Participant Creation field will be reset to zero, but additional bulk requests can be submitted as needed by following these steps again.

9.0 STEPS TO ADDING A TASK IN MODME

9.1 Background and Creating the Necessary Files

This outline is intended to help developers implement new experiment tasks in ModME, so some experience with coding will be necessary. Part of this process involves the creation of three documents needed to create the new task, and this overview will cover how the documents are used in the process and some common elements they all share. To demonstrate the process, the Situational Awareness task will be used as an example. All files and images are located in \ModME\static\ModME. The easiest way to create a new task is by looking at the existing tasks and tweaking them as needed. Making changes to any MAT-B_name_chart.js file ensures the most immediate feedback loop on the path to understanding.

Every task uses six files, three of which are shared by all tasks. These shared files are:

1) d3/d3.v3.min.js,
2) ModME/MAT-B_Styles.css
3) ModME/guiUtil.js.

At a high level, the other 3 files are as follows:

1) MAT-B_[taskName]_chart.js
    a) The physical representation of a task. Handles all movements, SVG drawing and more. Components are drawn and manipulated using D3.js
2) [taskName].js
    a) Creates initial SVG canvas and holds chart.when function to allow data to be saved to the database.
3) [taskName]Configurator.js
    a) Allows for task settings to be changed. Can adjust probabilities, speeds, etc.

These 3 files will be discussed in more detail below. Before getting started, there are several online resources that may be useful to reference:

- https://github.com/d3/d3-3.x-api-reference/blob/master/Quantitative-Scales.md#linear
- https://bost.ocks.org/mike/join/
- https://www.d3indepth.com/enterexit/

9.1.1 MAT-B_[taskName]_chart.js

This file is the chart itself that holds the task components. Note that the part of the file name in brackets, *taskname,* must be modified depending on the task to be added. The file runs in both the configuration page and when an experiment is run. The file begins with an initialize function and is run once when the chart has loaded.

```
//TODO check this
chart.x = d3.scale.linear().range(0,chart.w).domain(0, 100);

chart.y = d3.scale.linear().range(chart.h, 0).domain(0, 100);
```

These variables help position elements in the task in a way that is easier for the developer to understand and that scales well on different sized screens. ModME is using an older version of Data-Driven Documents Application Programming Interface (D3 API), so this code differs slightly from modern documentation which uses *d3.scaleLinear()*. Documentation for the older API is available at: https://github.com/d3/d3-3.x-api-reference/blob/master/Quantitative-Scales.md#linear.

There are listener arrays in every task chart document. The "when" function at the bottom of the document adds a function to one of these arrays and is called in the *taskName.js* file. When an event happens, functions in these arrays can be called to log the event.

The sample code below includes a *chart.alert.forEach* with the *generateEvent* function in the *MAT-B_AircraftCoordination_chart*. This then calls the *when("alert")* function in aircraftCoordination.js and logs the appropriate data to the database.

```
chart.alert.forEach(function(d){d({domID: "aircraftCoordination"});});
```

```
var usersAircraft = chart.base.append("g");
this.layer("usersAircraft", usersAircraft, {
      dataBind: function(data){…},
      insert: function(data){…},
      events: {
            "enter": function(){…},
            "update": function(){…},
            "exit": function(){…}
     }
});
```

Items are added to the chart in layers. Each layer contains a particular type of object. The layer added last will be on top. This means that when a location with overlapping layers is clicked on the top layer will be selected. For a detailed description of what each function in the layer does, it is recommended that the developer look at a good tutorial for the D3 API. Good resources explaining these functions include: https://bost.ocks.org/mike/join/ and https://www.d3indepth.com/enterexit/ . ModME will look a little different from these articles but the functions have the same purpose.

```
chart.defaults.generateAlert = function(){
    if(chart.data.distractor){
        return null;
    }
    //do stuff here

}
chart.alertGenerator(options.generateAlert || chart.defaults.generateAlert);
chart.beginAlert = function(alert) {…}
chart.alertEvent = function() {
    var alert = chart.generateAlert();if (alert) {
        chart.beginAlert(alert);
    }
    var timeInMillisecondsToNextAlert = chart.eventFunction();
    if (null === timeInMillisecondsToNextAlert)
        return; // no more events
    setTimeout(chart.alertEvent, timeInMillisecondsToNextAlert);

}
//startFunction is currently a misnomer.  It is used here as a number, not a
//function.
setTimeout(function(){setTimeout(chart.alertEvent, chart.startFunction);}, 1);
```

Most tasks in ModME include some sort of alert mechanism, a periodic event that requires participant response within an established timeframe. If a task requires that something happen continuously, then *setInterval* is used with or in place of *setTimeout*.

The *generateAlert* function must handle the task of being a distractor, in which case it should not return any alerts.

The *options.generateAlert* function is used if the task is set to send pre-programmed alerts. This function is created in the taskName.js file.

9.1.2 [taskName].js

This file creates the task when the experiment is run. The part of the file name in brackets should be replaced with the appropriate name for the task to be added, such as *situationalAwareness.js*.

```
var situationalAwarness_data = setup.Tracking.data;
```

This line takes the data from the configuration page and makes it available to the chart. A few objects will be added to it before the task is run. This variable can be useful to look at while debugging.

```
situationalAwarness_svg = GUIUtil.getGenericSVG(…)
```

This line creates the actual Scalable Vector Graphics (SVG) object that is added to the HTML file. This can be one of two sizes: 650 x 650 or 1300x650. The larger size is for tasks that are intended to run on the right side of the screen (Figure 62).

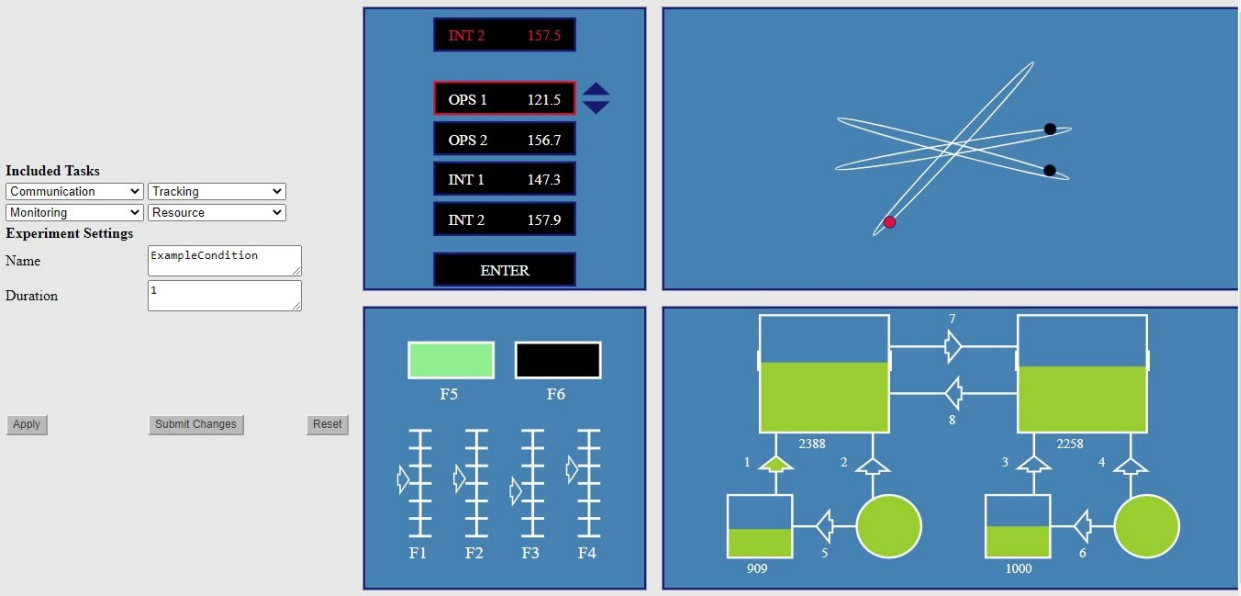


Figure 62. Comparison of Task Sizes (i.e., Left vs. Right) when Sized to 1300x650.

```
var situationalAwarness_chart = …
```

This line makes the chart easily available and can be useful to look at when debugging.

```
if (window.preprogrammedAlerts) {
} else{
}
```

This section handles event data reuse. This section can be copied and pasted from other tasks with only obvious name changes.

```
if(!situationalAwarness_data.distractor) {
} else{

}
```

This section changes what data are logged depending on whether the task is included as a distractor. If it is configured as a distractor no alerts should be created or logged. This section will change a little depending on the task. Some tasks, such as Tracking, require logging at periodic intervals using "*tick*". Others require logging mouse locations and add "*mouseMove*".

Because this file is run during experiments but not in the configuration page, it sometimes includes listeners that are not used in the configuration page. For example, the Resource task includes keyboard listeners at the end of *resource.js* that listen to keycodes. Testing keyboard inputs in the configuration page will not work. For this example, the feature must be tested in the experiment page. Other listeners, such as the click listeners, are included in the *MAT-B-Resource_chart.js* file and can therefore be debugged in the configuration page. Deciding which of these approaches is better depends on developer and researcher preferences.

9.1.3 [taskName]Configurator.js

This file starts the task in the Configuration page, thus allowing users to edit the data that will draw the chart. The part of the file name that is in brackets should be replaced with the appropriate name for the task being added, such as *situationalAwarenessConfigurator.js*.

The file contains click listeners for each layer of the task. These listeners modify the HTML elements of the page, showing or not showing elements based on what data need to be edited.

There are *task objects* and *object objects*. *Task objects* modify general settings within the task. *Object objects* modify the specific layer on which the participant clicked. Task objects can be modified by clicking on any layer in the task. It is common to modify the code for a task object in one layer but not in others.

Near the bottom of the file is the default data for the task. These are the data actually displayed by the task. A screenshot of the configurator panel is shown in Figure 63. In this example, the cursor is over the F5 button of the monitoring task and has been clicked. This displays the configuration parameters for the Monitoring task that can be modified (e.g., probability of changing colors, time to timeout, keyboard binding, etc). Other objects in the task can be interacted with and modified using the same method.

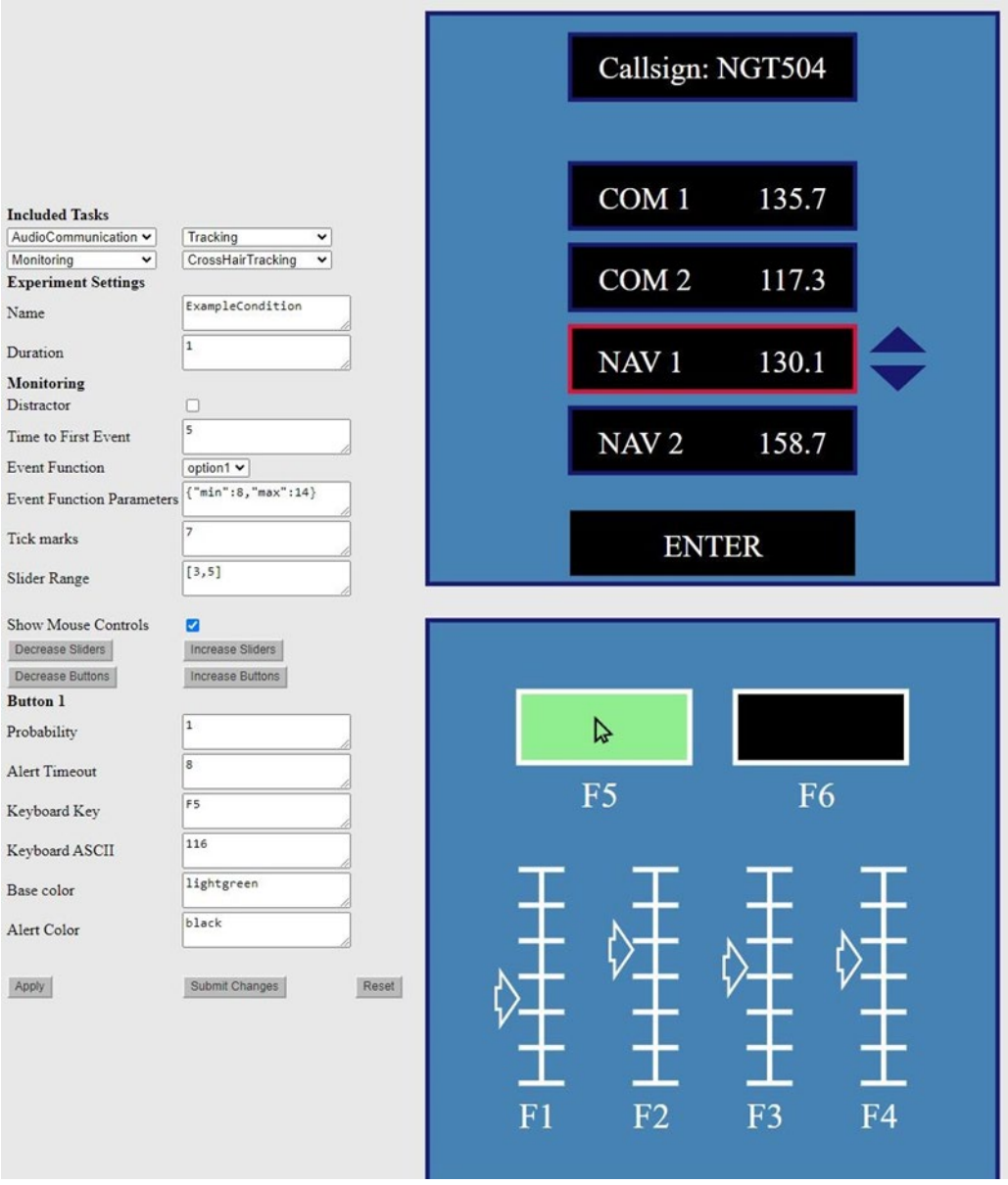


Figure 63. Example of the Configurator Panel in Task Setup

9.2 Adding Task to ModME

Once the files are created, the task has to be added to the ModME database. There are two ways of doing this.

The first is to add to the configuration page manually. Note that this will **not** permanently add the task to ModME. If the database is deleted and rebuilt, then the task will have to be re-added.

The second way is to edit the migration to add the task to the database automatically when the task is rebuilt. The second method is recommended and will be explained here.

The file that will need to be modified is *ModME/migrations/0003_add_default_tasks.py* . Outlined below are the necessary steps:

1) Create the task object.

```
situationalAwarness =
Task.objects.get_or_create(fileName='ModME/situationalAwarness.js',
taskName='SituationalAwarness',
configurator='ModME/situationalAwarnessConfigurator.js')[0]
```

2) Add the other required files to the database in association with the task.

```
File.objects.get_or_create(name='ModME/MAT-
B_CrossHairTacking_chart.js',task=situationalAwarness)
File.objects.get_or_create(name='d3/d3.v3.min.js', task=situationalAwarness)
File.objects.get_or_create(name='ModME/MAT-
B_Styles.css',task=situationalAwarness)
File.objects.get_or_create(name='ModME/guiUtil.js', task=situationalAwarness)
```

3) Add the task to the list of tasks to be removed if the migration was reversed.

```
Task.objects.filter(taskName__in=[…] ).delete()
```

4) Shutdown the server and delete and rebuild the database.

Please be aware that changes to the database will make old versions of ModME incompatible with the new version and play close attention to version details indicated online.

APPENDIX A COMPLETE PARAMETER LIST

- **General Settings for Condition Administration**
    - Location: Indicates current page while logged into the Administration Site. In the screen shot above, the current location is Home > Modme > Conditions.
    - Action Selection: Drop-down field from which to select "actions" that may be taken on the listed conditions. At the time this document was prepared, the only action available is to delete items. To perform an action, select a condition in the Action Item Selection Column, select the desired action, and then click on the button labeled *Go*. A confirmation pop-up will appear on the screen asking for confirmation, at which point the requested action will be completed.
    - Action Item Selection Column: These are the check boxes in the left column of this table. Multiple conditions will need to be selected in this column before accessing the Action Selection tool described previously.
    - Condition ID Column: Lists the IDs of each condition created in the account.
    - Name Column: Lists the name assigned to each condition.
    - Instructional PDF Column: This is the full file path and name of the instructional document that is linked to the condition. Note, these are not instructions for completing the individual tasks but rather are instructions specific to the specified condition and used to explain each step in greater detail to participants.
    - Configuration Column: Direct links to each of the existing conditions where modifications can be made or settings verified.
    - Add Condition Button: Click here to add another condition. Doing this lists the conditions that are included. Please refer to Section 5.2, "Creating a New Condition", in this document for additional detail.
    - Instance Counter: Indicates ow many conditions are currently in the corresponding instance of ModME.
- **General Settings When Creating a New Condition**
    - Name: Enter a meaningful name for the condition. Any alphanumeric text may be entered here. Names are most effective when they are meaningful, distinct, and short.
    - Length of Experiment (minutes): This is the length of time the condition will run (in minutes). For instance, if "5" is entered in this field, then this condition will run for 5 minutes.
    - Instructional PDF: Instructions for the specified condition, saved as a PDF. These instructions will be presented to the participant at the start of each condition. This step may be omitted if not relevant to the condition; see "Skip Instructional PDF" below.

- Skip Instructional PDF: If selected, the instructions for a PDF will not be presented to participants, even if a PDF file has been attached to the condition. NOTE: This field must be updated before removing a PDF if one has been attached.
- Condition Ending Message: The intent for this field is to present a message to participants after completing the task condition. Possible examples might include telling participants to take a 5-minute break or to tell them the study is complete and thank them for their time.
- Surveys: This is used to append another survey at the end of each condition. The National Aeronautics and Space Administration Task Load Index (NASA TLX) has already been built into the system, but researchers may add other surveys as needed.

- **Condition Configuration**
  - Task Selection, 1 – 4: Field to select which task will appear in the top left quadrant.
  - Name: Name of the condition, which is auto filled to match the name of the condition selected on the previous page.
  - Duration: The desired duration for the condition, in minutes, should be entered here. Note, to optimize data storage, conditions should be no longer than 20 minutes. The default duration for all conditions is 10 minutes.
  - Apply Button: Applies changes made to configuration settings for the condition.
  - Submit Changes Button: Saves the configuration changes made and returns the participant to the main Conditions page.
  - Reset Button: Will revert changes made to the condition.
- **Audio Communication Task**
  - Parts of the Task
    - *Call sign Box*: This box displays the participant's call sign.
    - *Communication Channels*: A set of boxes, each with a name and current frequency. The participant can select different channels using the up and down arrow keys or by clicking on a channel. When the box is selected, the box's outline will be red.
    - *Frequency Change Arrows*: The participant can change the frequency of the current box by using these arrows or with the left and right arrow keys.
    - *Enter Box*: When the participant has made the desired change to the frequency, they can submit their change by clicking on this enter box or pressing the "Enter" key. After a change has been submitted, the outline of the box will turn green until the participant needs to make another change.

- Configuration Options
    - *Distractor*: Checking this box turns the task into a distractor. This means that audio command will never belong to the participant's call sign and subsequently never require input from the participant. The task will only be there to provide audio and visual stimulus.
    - *Time to First Event*: Sets the number of seconds until the first audio command is given.
    - *Event Function*: This selects how audio commands will be dispersed. If option 1 is selected there will be a random amount of time between alerts within the min and max event function parameters. If option 2 is selected, then the alerts will be geometrically distributed with an average time set by the event function parameter *avg_wait*.
    - *Event Function Parameters:* This box contains the parameters used by the event function. If the random function (option 1) is selected, then the min and max times will be needed. If the geometric function is selected, then the *avg_wait* will be needed.
    - *Alert Timeout*: Determines the number of seconds the participant has to respond before the instruction times out.
    - *Files*: This box contains the audio list of files that the task will play as commands. The files should have names of the form Call sign_Channel_Freq.wav. If the call sign in the filename is equal to the participant call sign setting below, then the command will be considered one that the participant should respond to. Other commands should be ignored by the participant.
    - *User Call Sign*: The call sign displayed in the text box at the top of the task and is the call sign assigned to the participant for this session. It is also used to select which of the audio files belong to the participant. This is an important part of the experimenter playing the participant's call sign with a specific probability. Use of a keyboard to enter responses to other tasks presented in conjunction with this Tracking task will enable participants to continue tracking the satellites with a mouse.
    - *User Probability*: This is the probability (out of 10) that an instruction with the participant's call sign will be called. For example, if "4" is entered, then four out of the 10 possible instructions will be addressed to the participant using his or her call sigh, while the other six orders will be addressed to fictional call signs.
    - *Number of Events*: This setting is used if the Random Event Spacing box below is not checked. It will evenly space the alerts such that the desired number of alerts will be presented during the task.

        - *Random Event Spacing*: If this is checked, then the event function parameter will be used to distribute the alert; otherwise tasks will be evenly spaced to match the Number of Event setting.
        - *Show Mouse Controls*: If checked, the up and down arrows will display next to the selected channel. The arrows can be clicked to adjust the channel's frequency up or down.
        - *Decrease Channels / Increase Channels*: These buttons on the configuration page increase or decrease the number of channels.
        - *Channel Name*: This is displayed on the left side of the channel's box. It is also used to determine if the correct channel was selected after an alert.
        - *Probability*: Controls the probability of the specified channel being called in relation to the other channels included in the condition.
        - *Min Freq*: The minimum starting frequency of the channel.
        - *Max Freq*: The maximum starting frequency of the channel.

- **Communication Task**
    - Parts of the Task
        - *Instruction Box*: This box displays a channel and frequency. Periodically, the text will change the color to red which indicates an alert. After a predetermined amount of time, specified by the researcher, the alert will timeout and the text will change back to white.
        - *Communication Channels*: This is a set of boxes with names and frequencies. The participant selects different channels using the up and down arrow keys or by clicking on different channels. When a channel is selected, its outline will be red.
        - *Frequency Change Arrows*: Participants can control the frequency of the selected channel using either these arrows or the left and right arrow keys on a keyboard.
        - *Enter Box*: When the participant has made the desired change to the frequency, they can submit their change by clicking on this enter box or by pressing the "Enter" key.
    - Configuration Options
        - *Distractor*: Checking this box turns the task into a distractor, meaning that the instruction text will never turn red and the participant will never have to interact with the task.
        - *Time to First Event*: This is the number of seconds between the start of the task and the first alert.

- *Event Function*: This controls how alerts will be presented to participants.
    - Option 1 – Alerts will be presented at random intervals, within the established alert and event function parameters min and max.
    - Option 2 – Alerts will be presented at an interval set by the researcher using the event function parameter *avg_wait*.
- *Event Function Parameters*: This box contains the parameters used by the event function. If the random function (option 1) is selected, then the min and max parameters will be needed. If the geometric function is selected (option 2) then the *avg_wait* will be needed.
- *Alert Timeout*: Determines the number of seconds the participant has to respond before the instruction times out.
- *Frequency Minimum*: The lowest frequency that will be chosen for an alert event.
- *Frequency Maximum*: The maximum frequency that will be chosen for an alert event.
- *Show Mouse Controls*: If checked, the up and down arrows will be displayed next to the selected channel.
- *Decrease Channels / Increase Channels*: These buttons on the configuration page increase or decrease the number of channels.
- *Channels Name*: This is displayed on the left side of the channel's box.
- *Frequency Differential*: The max difference that an alert can be from the current frequency of a channel.
- *Probability*: The probability that the channel will be chosen. The probabilities of all the channels are cumulative. For example, if there are four channels with probabilities of "1", then the probability of any channel being selected is 1/4. However, if the probability of one of the channels is changed to "2", then the probability of that channel being selected is 2/5 and the probability of the other channels is 1/5.

- **Crosshair Tracking Task**
    - Parts of the Task
        - *Target:* This moves around randomly. It can be moved by a joystick, if configured, or by clicking and dragging using the mouse. When the target is clicked on or selected in the joystick it will turn red.
        - *Target Region:* This is a circle or rectangle. The program will log when the target moves outside of this region. The participant's goal is to keep the target inside of this region.

- Configuration Options
    - *Distractor:* Checking this box turns the task into a distractor. This means that the target will never exit the region and so the task will never require input from the participant. The task will still be visible as a visual stimulus.
    - *Use Joystick:* If the box is not checked, then the participant cannot control the target with the gamepad. When this box is checked, it allows the participant to control the target with the gamepad by holding down a button on the controller and moving joystick. Which button and axis on the controller are used can be configured in the Joystick Calibration settings below. Participants can click and drag the target with the mouse if this box is selected.

- **Monitoring Task**
    - Parts of the Task
        - *Buttons*: Each button has a base and alert color. The button starts off as the base color but will temporarily change colors to indicate an alert. When the button is in an alert status, the participant can change it back to its base color by either clicking on it or pressing a keyboard key associated with the button. The associated key is marked below the button.
        - *Sliders*: Below the buttons are four vertical lines with tick marks and an arrow that moves up and down the line. Ordinarily, the slider moves within a normal range but during an alert the slider moves outside of this range. When this happens, the participant can reset the slider by clicking on it or by pressing a keyboard key associated with the slider. The associated key is marked below the slider.
    - General Configuration Options
        - *Distractor*: Checking this box turns the task into a distractor. This means that the target will never exit the region and so the task will never require input from the participant. The task will still be visible as a visual stimulus.
        - *Time to First Event*: This is the number of seconds between the start of the task and the first alert.
        - *Event Function*: This setting is used to control how alerts will be presented over the course of the task.
            - Option 1 – Alerts will be presented at random intervals, within the established alert and event function parameters min and max.

- Option 2 – Alerts will be presented at an interval set by the researcher using the geometric distribution of alerts with an average time set by the event function parameter *avg_wait*.

- *Event Function Parameters*: This box contains the parameters used by the event function. If the random function (option 1) is selected min and max times should be entered. If geometric distribution is selected (option 2) then *avg_wait* should be entered.
- *Tick Marks*: This is the number of tick marks on each slider.
- *Slider Range*: This is the base range for the sliders when they are not in an alert state.
- *Decrease Sliders / Increase Sliders*: These buttons increase and decrease the number of sliders.
- *Decrease Buttons / Decrease Buttons*: These buttons increase and decrease the number of buttons.

o Configuring the Buttons

- *Probability*: This is the probability that this button will be chosen for an alert. The probabilities of all the objects, both buttons and sliders, are added up and one object is selected. For example, if there are four sliders and two buttons, all with probabilities of "1", then the probability that any one of these objects will be selected is 1/6. However, if one of the buttons has its probability changed to "2", then its likelihood of having an alert is 2/7 whereas the other objects have a likelihood of 1/7.
- *Alert Timeout*: This is the amount of time the participant has to respond to an alert.
- *Keyboard Key*: The text to be displayed under the button, used to control the key corresponding to each button.
- *Keyboard ASCII*: The JavaScript event keycode of the key to be associated with the button. Note that this is different from a standard ASCII value.
- *Base color*: The color of the button when it is not an alert (see Appendix B)
- *Alert Color*: The color of the button when it is an alert (see Appendix B).

o Configuring the Sliders

- *Probability*: This is the probability that this slider will be chosen for an alert. The probabilities of all the objects, both buttons and sliders, are added up and one object is selected. For example, if there are four sliders and two buttons, all with probabilities of "1", then the probability that any one object will be selected is 1/6. However, if one of the buttons has its

probability changed to “2”, then its likelihood of having an alert is 2/7 and every other object has a likelihood of 1/7.

- *Slider Interval*: The number of milliseconds for the slider to move one tick.
- *Keyboard Key*: The text to be displayed under the button, used to control the key corresponding to each button.
- *Keyboard ASCII*: The JavaScript event keycode of the key to be associated with the slider. Note that this is different from a standard ASCII value.

- **Resource Task**
  - Parts of the Task
    - *Main Tanks*: These tanks have a resource level represented by a green area and a number below the tank. Target range is indicated by the thicker bars on the sides of the tanks.
    - *Fuel Sources*: These circles do not have a set resource level but instead they represent an infinite resource.
    - *Switches*: These operate as gates and can be opened to allow resources to flow in the direction of the arrow. The participant can open a gate by clicking on it or pressing the number next to the switch. A green gate indicates a gate that is currently open, while a hollow gate means it is currently closed. Red gates are broken and can’t be opened by the participant which is called an alert.
    - *Reserve Tanks*: Resources are not pulled from these tanks unless the corresponding gates are open, allowing resource to flow out of it. They have a number below them to indicate their current resource level.
  - General Configuration Options
    - *Distractor*: Checking this box turns the task into a distractor, meaning that the switches will never break and the main tanks resource level will not decrease below the target level. No alerts will be generated, so the participant will never have to interact with the task.
    - *Time to First Event*: The number of seconds to the first alert after the task begins.
    - *Event Function*: This setting is used to control how alerts will be presented over the course of the task.
      - Option 1 – Alerts will be presented at random intervals, within the established alert and event function parameters min and max.

- Option 2 – Alerts will be presented at an interval set by the researcher using the geometric distribution of alerts with an average time set by the event function parameter *avg_wait*.

- *Event Function Parameters*: This box contains the parameters used by the event function. If the random function (option 1) is selected then min and max times should be entered. If geometric distribution is selected (option 2) then *avg_wait* should be entered.
- *Refresh Rate*: This sets how often the task refreshes. Each time the page refreshes, three things will happen:
    1. The task state is recorded.
    2. The main tanks decay an amount determined by their settings and refresh rate.
    3. The open switches transfer an amount of resource based on their settings.

- Configuring the Individual Tanks (accessible by clicking on the outline of the tank or the resource area)
    - *Max Resource*: The number of units in the tank when it is full.
    - *Starting Resource*: The number of units in the tank when the task starts.
    - *Decay Rate* (units/min): The number of units that the tank decays in a minute. By default, this value is "0" for the reserve tanks, but it can be changed.
    - *Target Range Minimum / Target Range Maximum*: Sets the target range of the task. They also set where the bold lines on the sides of the tanks are drawn to indicate the target range to the participant. These settings are not set for any tank other than the main tanks.
- Configuring the Individual Switches (accessible by clicking on the desired switch)
    - *Transfer Rate* (units/min): The number of units that the switch will transfer in a minute when it is on.
    - *Repair Time*: The number of milliseconds before a switch will repair itself after changing for an alert.
    - *Keyboard Key*: The letter that will be shown next to the switch.
    - *Keyboard ASCII*: On the switch, the ASCII value of the key that needs to be pressed to control the opening and the closing of the gate.
    - *Probability*: This is the probability that this switch will be chosen for an alert. The probabilities of all the switches are added up and one is selected. For example, if each of the eight switches has a probability of "1", then

the probability that any one will be selected is 1/8. However, if one has its probability changed to "2", then its likely hood of having an alert is 2/9 and every other object has a likely hood of 1/9.

- **Tracking Task**
    - General Configuration Options
        - *Distractor*: Checking this box turns the task into a distractor. This means that the satellites will stay black so there is never any need for the participant to interact with the task.
        - *Time To First Event*: Sets the number of seconds until the first alert.
        - *Event Function*: This setting is used to control how alerts will be presented over the course of the task.
            - Option 1 – Alerts will be presented at random intervals, within the established alert and event function parameters min and max.
            - Option 2 – Alerts will be presented at an interval set by the researcher using the geometric distribution of alerts with an average time set by the event function parameter *avg_wait*.
        - *Event Function Parameters*: This box contains the parameters used by the event function. If the random function (option 1) is selected, then the min and max parameters must be specified to control the number of seconds from the start of one alert to the beginning of the next. If the geometric function is selected (option 2) then the desired average amount of time (i.e., *avg_wait*) must be specified as this controls the average number of seconds from the start of an alert to the start of the next one.
        - *Refresh Rate*: Sets the number of milliseconds between each system state recording.
        - *Decrease Paths / Increase Paths*: These buttons increase and decrease the number of paths by one. Paths are added to the top left corner of the task box with a satellite.
    - Configuring the Satellites and Paths (accessible by clicking on the desired item)
        - *Points*: This box contains an array of points that determine where the path is. Each point has an x and y property. These represent the position of the point with (0,0) being the bottom left corner and (1,1) being the top right corner.
        - *Path Interval*: The number of seconds it takes the satellite to complete one trip along all the points.
        - *Satellite Radius*: The radius of the satellite.

- *Probability*: The probability that this satellite will be chosen for an alert. The probabilities of all the paths are added up and one is chosen. For example, if there are four paths with a probability of "1", then the probability of any path being selected is 1/4. However, if the probability of one path is changed to "2", then the probability of the selected is 2/5 and the probabilities of the other paths are 1/5.
- *Path Width*: The width of the path—note that a value of "0" will make the path disappear.

- **Joystick Calibration:**
  - General Configuration Options
    - *scale*: Controls how sensitive the joystick is. If this value is larger, then the target will move more with a given amount of joystick input. If this value is smaller, then the target will move less with the same amount of input. A value of "0" will prevent the target from moving in response to movement of the joystick.
    - *xAxis, yAxis*: Control which axis on the controller move the target. *xAxis* specifies what axis move the target left and right, and *yAxis* specifies what axis move it up and down. There are various tools available to determine which axis corresponds to a particular button on the controller used, such as https://gamepad-tester.com/.
    - *trigger*: Determines what button on the controller needs to be pressed to take control of the target. Mapping of the buttons can be determined by using an external gamepad tester (e.g., https://gamepad-tester.com/).
    - *zeroX, zeroY*: Sometimes a particular controller will need to be calibrated, such as if the participant notices that movement is necessary to hold the target in place. When this happens consistently, *zeroX* and *zeroY* should be used. They control what joystick position is 0 for each axis.
    - *minX, maxX, minY, maxY*: If the controller is more sensitive in one direction, then these settings are used. A normal controller range is from -1 to 1 on each axis, with "0" signifying no input. It may be that a particular controller is not using this range. Such a controller may only go from -0.7 to 0.8, but the values can be changed to correct for this.
    - *Region Type*: This controls the shape of the region. There are two possible settings: circle or rectangle. If any other setting is entered, the task will default to using a rectangle. Please note that the region dimensions may look different in the actual experiment page. A circle may look more like an oval, or a rectangle may appear more similar to a square.
    - *Jitter Speed*: This controls how fast the target randomly moves. A larger value makes it move faster and a smaller value make is move slower. A

value of “0” will make it not move, which can be useful when calibrating controller settings.

- *Cursor Width and Height*: This controls the size of the target.

- Configuration Options in the SQL Database

  - *rangeX, rangeY*: These influence all of the other appearance settings that allow for precise control tick marks and target region placement in a way that is less influenced by screen dimensions. The idea is that the experimenter can divide the task box into a certain number of units. By default, rangeX and rangeY are 8, indicating the box is eight units across left and right along with up and down. This creates a coordinate system that the other appearance settings can use. Coordinate (0, 0) is the top left corner. For example, with default rangeX:8 and rangeY:8, making the center of the box (4, 4) since this is halfway to the top and bottom along with the left and right edges.

  - *targetRegion*: This is an array that only contains one object, radius, which controls the radius of the region the participant is trying to keep the target in. The region will always be centered in the task box. If *regionType* is set to “circle”, then a circle will be drawn with this radius; otherwise, a rectangle will be drawn with side radius units from the center.

  - *crossHairMarks*: An array that contains the location of each of the tick marks. Each of the *crossHairMarks* objects contains a *locX* and *locY* which are the number of units from the top-left corner of the task box with the units defined by the *rangeX* and *rangeY* settings.

APPENDIX B VALID CSS COLORS

To see each, visit https://w3schools.sinsixx.com/css/css_colornames.asp.htm

Color Name	Color HEX
AliceBlue	#F0F8FF
AntiqueWhite	#FAEBD7
Aqua	#00FFFF
Aquamarine	#7FFFD4
Azure	#F0FFFF
Beige	#F5F5DC
Bisque	#FFE4C4
Black	#000000
BlanchedAlmond	#FFEBCD
Blue	#0000FF
BlueViolet	#8A2BE2
Brown	#A52A2A
BurlyWood	#DEB887
CadetBlue	#5F9EA0
Chartreuse	#7FFF00
Chocolate	#D2691E
Coral	#FF7F50
CornflowerBlue	#6495ED
Cornsilk	#FFF8DC
Crimson	#DC143C
Cyan	#00FFFF
DarkBlue	#00008B
DarkCyan	#008B8B
DarkGoldenRod	#B8860B
DarkGray	#A9A9A9
DarkGreen	#006400
DarkKhaki	#BDB76B

Color Name	Color HEX
DarkMagenta	#8B008B
DarkOliveGreen	#556B2F
Darkorange	#FF8C00
DarkOrchid	#9932CC
DarkRed	#8B0000
DarkSalmon	#E9967A
DarkSeaGreen	#8FBC8F
DarkSlateBlue	#483D8B
DarkSlateGray	#2F4F4F
DarkTurquoise	#00CED1
DarkViolet	#9400D3
DeepPink	#FF1493
DeepSkyBlue	#00BFFF
DimGray	#696969
DodgerBlue	#1E90FF
FireBrick	#B22222
FloralWhite	#FFFAF0
ForestGreen	#228B22
Fuchsia	#FF00FF
Gainsboro	#DCDCDC
GhostWhite	#F8F8FF
Gold	#FFD700
GoldenRod	#DAA520
Gray	#808080
Green	#008000
GreenYellow	#ADFF2F
HoneyDew	#F0FFF0
HotPink	#FF69B4
IndianRed	#CD5C5C
Indigo	#4B0082

Color Name	Color HEX
Ivory	#FFFFF0
Khaki	#F0E68C
Lavender	#E6E6FA
LavenderBlush	#FFF0F5
LawnGreen	#7CFC00
LemonChiffon	#FFFACD
LightBlue	#ADD8E6
LightCoral	#F08080
LightCyan	#E0FFFF
LightGoldenRodYellow	#FAFAD2
LightGrey	#D3D3D3
LightGreen	#90EE90
LightPink	#FFB6C1
LightSalmon	#FFA07A
LightSeaGreen	#20B2AA
LightSkyBlue	#87CEFA
LightSlateGray	#778899
LightSteelBlue	#B0C4DE
LightYellow	#FFFFE0
Lime	#00FF00
LimeGreen	#32CD32
Linen	#FAF0E6
Magenta	#FF00FF
Maroon	#800000
MediumAquaMarine	#66CDAA
MediumBlue	#0000CD
MediumOrchid	#BA55D3
MediumPurple	#9370D8
MediumSeaGreen	#3CB371
MediumSlateBlue	#7B68EE

Color Name	Color HEX
MediumSpringGreen	#00FA9A
MediumTurquoise	#48D1CC
MediumVioletRed	#C71585
MidnightBlue	#191970
MintCream	#F5FFFA
MistyRose	#FFE4E1
Moccasin	#FFE4B5
NavajoWhite	#FFDEAD
Navy	#000080
OldLace	#FDF5E6
Olive	#808000
OliveDrab	#6B8E23
Orange	#FFA500
OrangeRed	#FF4500
Orchid	#DA70D6
PaleGoldenRod	#EEE8AA
PaleGreen	#98FB98
PaleTurquoise	#AFEEEE
PaleVioletRed	#D87093
PapayaWhip	#FFEFD5
PeachPuff	#FFDAB9
Peru	#CD853F
Pink	#FFC0CB
Plum	#DDA0DD
PowderBlue	#B0E0E6
Purple	#800080
Red	#FF0000
RosyBrown	#BC8F8F
RoyalBlue	#4169E1
SaddleBrown	#8B4513

Color Name	Color HEX
Salmon	#FA8072
SandyBrown	#F4A460
SeaGreen	#2E8B57
SeaShell	#FFF5EE
Sienna	#A0522D
Silver	#C0C0C0
SkyBlue	#87CEEB
SlateBlue	#6A5ACD
SlateGray	#708090
Snow	#FFFAFA
SpringGreen	#00FF7F
SteelBlue	#4682B4
Tan	#D2B48C
Teal	#008080
Thistle	#D8BFD8
Tomato	#FF6347
Turquoise	#40E0D0
Violet	#EE82EE
Wheat	#F5DEB3
White	#FFFFFF
WhiteSmoke	#F5F5F5
Yellow	#FFFF00
YellowGreen	#9ACD32

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

ASCII	American Standard Code for Information Interchange
cd	MacOS command for "change directory"
CSS	Cascading Style Sheets
CSV	Comma-separated Values
D3 API	Data-Driven Documents Application Programming Interface
HTML	Hypertext Markup Language
ICD	Informed Consent Document
ID	Identification
IP	Internet Protocol
IRB	Institutional Review Board
ls	MacOS command for "list files"
macOS	Macintosh Operating System
MATB	Multi-Attribute Test Battery
ModME	Modular Multi-tasking Environment
MTurk	Amazon's Mechanical Turk
NASA TLX	National Aeronautics and Space Administration Task Load Index
PDF	Portable Document Format
PI	Principal Investigator
PII	Personally Identifiable Information

SQL	Structured Query Language
SVG	Scalable Vector Graphics
USB	Universal Serial Bus
VE	Virtual Environment