MassMobile Deployment Guide

v. 1.1, 4/23/14

About massMobile

massMobile enables large groups of people to participate in live performance experiences, in real time, using most commodity web browsers on common computing devices (e.g. smartphone, tablet, etc.) connected to any wide-area network (3G, 4G, WiFi, etc.).

massMobile supports the following devices and browsers:

- iOS devices.
- Android devices.
- Chrome, Firefox, and Safari on Mac and Windows.

At this time, Windows Phone, Blackberry, and IE are not supported.

massMobile consists of the following components:

- An HTML5 / javascript mobile web client, hosted on Heroku.com.
- MongoDB, a scalable cloud database, hosted on Heroku.com.
- A Node.js server for handling queries from mobile and desktop clients and calculating data and statistics, hosted on Heroku.com.
- A Java desktop client API for interacting with the database from a Windows or Mac laptop (e.g. to configure projects, pull down real time data, and develop audio/visual responses to the data in real time).
- A Max/MSP desktop client API that wraps the Java API, for those who prefer to use Max/MSP instead of Java directly.

To use massMobile with your project, you need to follow these steps:

1. Create and configure a new massMobile project with Max/MSP.
2. Access your massMobile project through a supported web browser.
3. Write Java or Max/MSP software to manage configuration and pull data for your project.
4. When you are ready to use massmobile in a public performance setting, create and configure a free account on Heroku, and deploy massMobile to your Heroku account instead of to our sandbox.

What you do with the audience data is up to you. In past work with massMobile, we have used it to generate electroacoustic sound, real-time music notation or performance instructions for instrumental musicians, video animations, and lighting cues.

The following sections walk through using massMobile in detail.

STEP 1: Create and configure a new massMobile project.

In massMobile, a project represents a particular user interface configuration of the HTML5 application and is usually specific to a particular artistic work.

To create and configure your massMobile project, use massMobile's Max/MSP helper app. If you do not already have Max version 6 or higher, download Max from http://cycling74.com/downloads/. Max is commercial software that comes with a fully-functional 30-day demo. If you prefer, you can instead download the free Max Runtime application at http://cycling74.com/downloads/runtime/. This will let you run the massMobile helper app but not create new software in Max/MSP.

Then open massMobile/max/massmobile.maxhelp in Max or Max Runtime and follow these steps:

1. Point the server URL (yellow box "1") to our MassMobile SandBox at http://massmobile.heroku.com unless you have created your own Heroku account (see Step 4, below), in which case you should instead enter http://<appname>.herokuapp.com as the server address.
2. Hit the "getProjectList" button to see the massMobile projects on your account. (Unless you've created projects before, the list will remain empty.)
3. Press the "startNewParamList" button. (Advanced users: If you have configured a massMobile project in your account previously and if you want to keep audience data stored in the same table on the server, then you can instead press the "getCurrentParamListName" button.)
4. Create a new project by typing in its name and hitting the "add" button in the "create new project area.
5. Select your new project in the project list.
6. Add one or more parameters to your project. In massMobile, a parameter represents a user interface tab of the mobile interface, such as a 1D slider, a 2D slider, a voting widget, a text-messaging widget, or a drawing widget. To add a parameter, give it a name, pick its type, and hit the add button.
7. Select each parameter in the parameter list and configure its options in the config list. To configure each option, click on it and edit its value in the edit area below. A full description of the configuration options is in an appendix at the end of this document. To make sure a parameter is active and will be displayed in the mobile client, set its "active" config option to 1.
8. When you are done, hit the updateProject button to save your project to Heroku.

STEP 2: ACCESS YOUR MASSMOBILE PROJECT ONLINE

Now that you've created your massMobile project, you are ready to use the audience participation application from any supported web browser or mobile device.

Web Client URL

Follow this pattern to create a valid URL for your massMobile application:

http://<appname>.herokuapp.com/public/MassMobile_web_client.html?projId=<projid>&userId=<userid>

=== optional ===

&debugMode=<debugMode>

&pollRateRaw=<pollRateRaw> (and/or) &pollRateAggregate=<pollRateAggregate>

<appname>	The name of the app that you created.
<projId>	The project ID you want to access. In the Max/MSP helper app, this shows up as “selected project ID” when you select a project from the project list.
<userId>	The user ID for special identity. Normally, this should be set to 0 so that a new random ID is generated for each user. If, however, you want to set a specific user ID for one or more users, so that you can later identify the data coming from a specific user, you can set it here.
<debugMode>	This is optional. Setting it to any number other than 0 will show the debugging console in the client app.
<pollRateRaw> <pollRateAggregate>	Optional. Set the polling rate for raw / aggregate data feedback. It can take an integer value in millisec. The default rate is 1000ms (1 polling per second). The minimal is 100ms, although it is not recommended to set it too low for mobile devices.

For example, a URL could be...

http://massmobile.herokuapp.com/public/MassMobile_web_client.html?projId=5&userId=0

Take a moment to create the correct URL for your massMobile project now and verify that you can access it. Note that when you change the configuration of your project in the Max/MSP helper app, it will automatically update in your web browser within a few seconds. This applies only to changes in configuration settings; the addition or deletion of an entire parameter (i.e. user interface tab) requires refreshing the browser window to update.

Making a QR code and short url for your massMobile project

It is unreasonable to expect anyone to type in these long URLs, so you need to create a short

URL and a QR code to make this process easier for users:

1. Use a free QR code generator such as http://qrcode.kaywa.com to generate a QR code that mobile devices can scan to access your massMobile project.
2. Use a free short URL generator such as https://bitly.com to generate a short URL to access your massMobile project. We encourage you to create a custom URL that is relevant to your project and easy to type.
3. Publicize your QR code and URL to audiences at your event through flyers, program booklets, and/or video projections.

STEP 3: Write your software in Java or Max/MSP.

The last step in creating your massMobile project is to write some software that takes the audience data from Heroku in (close to) real time and does something interesting with it!

You can write your software in either Java or Max/MSP. The two APIs are functionally equivalent, so use whichever you prefer.

Using the Java API

Materials for Java are in the massMobile/java/ folder. The doc/ subfolder includes the Javadocs, the example/ subfolder includes a simple example of using the API. When writing your application, be sure to include the JAR file in the root level of the directory.

When writing your application, you'll want to implement the MMClient interface so you can get notifications when data comes through asynchronously from Heroku. Most of your interaction with the API will be through the MMServerComm class. See the sample application for details.

Using the Max/MSP API

Materials for Max/MSP are in the massMobile/max/ folder. When creating a new patch, you must copy over all of these files to the same folder in which you create your patch so that the dependent patch files and Java JAR files can be found by Max.

massMobile.maxhelp is the first place to look for API documentation and sample code. The messages tab explains each of the messages you can send to the massMobile object, and the Documentation button on the main tab explains all of these options in greater detail. The 2D slider tab has a simple example of visualizing audience data from a 2D slider widget.

The massMobile/max/ folder includes a couple more example patches:

- Drawing.maxpat shows how to create a patch with a customized configuration interface to enable quick modification of the audience interface during the performance. (This patch does not actually query audience data from Heroku.)
- TeamWork.maxpat shows another example of visualizing data from the 2D slider, similar to the 2D Slider tab on the help patch.

- SaxophoneEtudes.maxpat shows another example of working with 2D slider data to shape a live performance.

STEP 4: When you are ready to use massMobile in a public performance, create and configure your own free account on Heroku.

What is Heroku?

Heroku is a robust, schemaless cloud database that helps massMobile scale to extremely large audiences. It is used by major companies such as Facebook, GitHub, and TED. Under the hood, massMobile uses MongoDB's API to store and query data and its Node.js architecture to calculate statistics on audience data. After performances, you can use Heroku's web console to export data from your performance for further analysis.

Why do I need my own account?

When you use our massMobile sandbox on Heroku, anyone can access your projects and change their configurations. You are also sharing a single server resource with everyone, so if multiple people are using the sandbox simultaneously, the server may slow down.

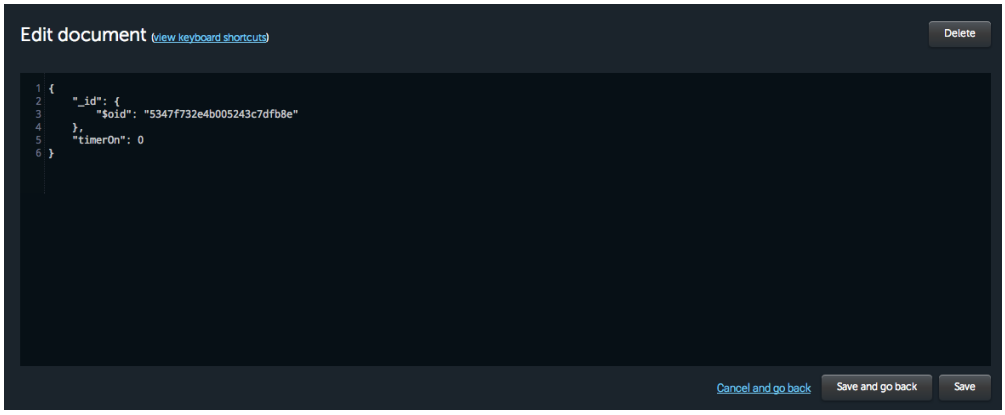
Having your own Heroku account also gives you access to audience participation data through the MongoDB web console, so you can export and analyze this data to learn more about how audience members participated in your event.

Create a Heroku Account and an Application with MongoLab Service

1. Visit Heroku.com and create a new user account. Their free service plan is typically sufficient for using massMobile.
2. Download and install Heroku ToolBelt (https://toolbelt.heroku.com/) which we will use for deploying massMobile on Heroku.
3. In the Dashboard of Heroku.com, create a new app (with whatever name you like). Note the application name and the Git command for later use (Deploy-2).
4. To use a MongoDB-based database, select "Get Add-ons", choose the "MongoLab" service with SandBox (free) plan, then select the application name that you have just created.
    - You may be asked to "verify the billing information" by entering credit card information before you are able to add MongoLab, and you may have to try to add it to the application again after the login and billing info have been entered. Heroku will not charge you as long as you remain within the free tier of services.
5. The following steps are IMPORTANT to properly configure your database and make them run without problem:
    - Click on the MongoLab SandBox you just created and you will be navigated to your linked MongoLab settings page. Take note of your Database name. (e.g.

"heroku_app12345678") Then, find the URI of your mongo db, which should be provided on the top of the landing page, in the form of: mongodb://<dbuser>:<dbpassword>@YourAppNumber.mongolab.com:yourportnumber/YourDatabaseName

- In the same page, add an empty collection with the name "**CurParamList**", an empty collection with the name "**ProjectList**", and an empty collection with the name "**StatsTimer**", and an empty collection with the name "**StatsRecord**".
- In the "**StatsTimer**" collection, manually add a document with a JSON pair "timerOn":1 and save it, like the picture shown below. In the "**StatsRecord**" collection, manually add an empty document. Simply click the "Add document" and save will do.



- Click on the Users tab in the same page and add a database user. Be sure to remember the username and password.

Deploy massMobile to Your Heroku App

Heroku hosts three parts of massMobile: the cloud data store (through MongoLab), the server-side code (through Node.js), and static HTML / javascript content to run the mobile web client. To upload these components and use massMobile, please follow these steps:

1. After installing the Heroku ToolBelt, open the Terminal / Git Bash and type "heroku login". You will be prompted to login with your email, and to choose or generate a "public key" (an SSH key). Type Y if this is the first time.
    a. On Windows Command Prompt, trying to generate an SSH key may result in the missing "ssh-keygen" error. If this is the case, please use the Git Bash program that is installed along with Heroku ToolBelt.
    b. In case you encounter other issues with SSH key, such as permission problems, please refer to https://devcenter.heroku.com/articles/keys
2. Create a local git repository by cloning the Heroku app using the git URL. In Terminal, set the current directory to the destination (e.g. "cd <path to the desktop>"), and enter "git clone git@heroku.com:<app name>.git -o heroku". The app name should be the same as the app name you used in step 3 of the previous section.

3. If you have not done so already, download the massMobile ZIP distribution.
4. Copy the contents of the “server” folder from the massMobile ZIP distribution into the root level of your local git repository. For example, if my Heroku application were called “mm” then I would copy the contents of the “server” folder into the “mm” folder on my computer.
5. Find **mydb.js** file from the files you copied. Do the following modifications:
    a. change line 2 to the MongloLab domain and the port number you found in Step 5 above. Which could look like this:

    ```
    server = new mongodb.Server('ds012345.mongolab.com', 12345, {
    ```

    b. change line 6 to the MongoLab YourDatabaseName found in Step 5 above. Which would be like:

    ```
    db1 = new mongodb.Db('heroku_app12345678', server);
    ```

6. Similarly, find **web.js** file from the copied files.
    a. Change line 29 to the username and password of the database user you created in Step 5 above. Which can be like:

    ```
    mydb.authenticate(db, 'yourusername', 'yourpassword', function(err, coll) {
    ```

7. In Terminal, set the current directory to the app folder (cd <dir name>), then type the following to update the repository. (You may be prompted to set your global git email and name for the first time.)
    a. git add .
    b. git commit -m “initial commit”
8. Deploy the app by typing “git push heroku master” in Terminal / Git Bash. The process may take a few minutes to complete.

Then be sure to reference your new Heroku URL when you use the Max or Java APIs and to provide an updated participation URL to audiences instead of using the massmobile.heroku.com sandbox.

Getting Help

Please don’t hesitate to contact us with questions or feedback on massMobile by filling out this contact form:

http://distributedmusic.gatech.edu/jason/contact.html

We hope you have fun using massMobile and look forward to hearing about your project!

APPENDIX A: PARAMETER CONFIGURATION OPTIONS

Here are details on the configuration options for each user interface parameter currently

available in massMobile.

PROJECT OPTIONS (available only at the project, not the parameter level)

- Project Info: This is the welcome message displayed when the web client loads. Enter as plain text or HTML.

GENERAL OPTIONS (available for all user interface parameters, but note that not all parameters implement all of these options)

- raw: The web client should poll raw data from Heroku so it can display all audience data on the device. (0 or 1)
- aboutText: Text displayed on the parameter's tab. (string)
- aggregate: The web client should poll aggregate data (i.e. statistics describing the audience data) for display on the device. (0 or 1)
- aggregatePollRate: The rate at which web clients should poll for aggregate data. (int in milliseconds)
- helpText: Text displayed on the parameter's tab. (string)
- displayText: Test displayed on the parameter's tab. (string)
- active: Determines whether the parameter's tab is displayed on the web client. (0 or 1)
- projectVersion: the version of the project's configuration, so that web clients know when to update their interface (integer) (never modify this directly, it is incremented automatically when calling updateProject).
- url: The short URL for audience members to load the web client (string).

SLIDER_1D:

- centerLabel: A string to be displayed in the center of the slider. (string)
- rightLabel: A string to be displayed on the right of the slider. (string)
- leftLabel: A string to be displayed on the left of the slider. (string)
- initialSliderVal: The intial value for the slider. (float, 0 to 1).
- verticalMode: Aggregate data is displayed in a format that is aligned to slider range (0 or 1).

SLIDER_2D:

- bottomLabel: A string to be displayed on the bottom of the slider (Y axis low). (string)
- topLabel: A string to be displayed on the top of the slider (Y axis high). (string)
- leftLabel: A string to be displayed on the left of the slider (X axis low). (string)
- rightLabel: A string to be displayed on the right of the slider (X axis high). (string)
- bkndImage: The url for an image file to be displayed behind the 2D slider (e.g. http://networkmusic.gatech.edu/massmobile/xyVolSpe.png). (url)
- initial2DSliderVal: The initial x and y values for the slider (two floats separated by a space, each 0 to 1).

VOTE:

- choices: A JSON list of the voting choices. (0:choice1 1:choice2 2:choice3)
- multiVote: Determines whether users can select more than one voting choice simultaneously. (0 or 1)

TEXT:

- charLimit: Character limit on the length of text entered. (int)

DRAW:

- performerMessage: A text message displayed in the audience interface. (string)
- limitUserDrawing: Whether to limit users to drawing a specified number of shapes per time period. (0 or 1)
- periodLength: the length of the period, in seconds, for drawing limits. (int)
- numShapesPerPeriod: the number of shapes each user may draw per time period before drawing is disabled for them. (int)
- countdownDuration: Duration, in milliseconds, of a countdown timer to show the length of the performance. (int)
- countdownStart: Start time, as ISO time string, for the countdown timer to being. (string)

PALETTE: (The palette can only be used in conjunction with DRAW.)

- drawParamId: The id of the DRAW parameter which the palette should control.
- numOfShapesEnabled: The number of shapes enabled on the palette (the rest will be greyed out). (int)
- shapeNames: A list of the strings to display with each shape, e.g. “bass:piano:sax:trumpet”.
- paletteAnnotations: A list of the strings to display with other elements in the drawing palette, e.g. “loudness:instruments:articulation:note length”
- numOfShapesShownUp: The number of shapes to display on the palette. (int)

APPENDIX B: MASSMOBILE DATABASE AND API OVERVIEW

Database Architecture

You don’t generally need to look at the database in Heroku directly, but here are details on how it is set up. To see the Heroku database, go to your app’s dashboard on Heroku.com and navigate to the MongoLab add-on page. To export your data, go to “Tools” and follow the Terminal command instruction.

- The ProjectList table stores information on the configuration of each project in your account; each row corresponds to a single project. The widgets in use and the

configuration of each are stored as JSON data in the parameters column. The versionNumber column keeps track of changes to this configuration so that it can propagate to all of the web clients. (The web clients periodically check the versionNumber of the project with Heroku; when the versionNumber is greater than their local version, they update their configuration accordingly. You can test this by making a change in a project's configuration in Max/MSP, hitting the updateProject button, and seeing the corresponding change in the mobile web client within a few seconds. The version number is incremented automatically by our APIs when config changes are pushed to Heroku.)

- The ParamList table stores every input (e.g. slider click, text entry, vote, etc.) received from the mobile web client. Each input is stored in a separate row. The data itself is stored as JSON data in the keyValues column (since some input actions may create more than one value, e.g. x/y of a touch).
- The CurParamList table stores the name of the current ParamList in use. In order to prevent ParamList tables from growing arbitrarily large and to keep audience data from different events separate for archival and analysis purposes, you can create a new ParamList for each performance or event you do. This is supported through the startNewParamList method in the Max/MSP and Java APIs. When you select this method, a new ParamList table is created with a timestamp appended to its name (e.g. ParamList_1366034010416). The name of this table is stored in CurParamList. The web client and the Java and Max/MSP APIs then grab data from this new ParamList. (The old data remains archived under the previous ParamList table name.)

Java and Max/MSP API Architecture:

The details of the API architecture are covered in the JavaDocs and Max/MSP help documentation, respectively, but here is a brief overview of the types of functionality supported.

- Project Configuration:
    - getProjectList shows all available projects in your account, setCurrentProject will load one by id, and createNewProject will add a new one.
    - Each user interface widget / tab in a project is called a parameter. You can list, add, and remove these using getParameterList, removeParameter, and addParamToCurrentProject. You can configure a particular parameter using getParameterConfigs and setParamConfigs.
    - You can modify the text displayed on a project's welcome screen using getProjectInfo and setProjectInfo.
    - To save a new configuration to the server, always call updateProject.
- Uploading Audience data:
    - Posting user input data to the server is straightforward with postData.
- Polling audience data from the server:
    - You can set the poll rate in milliseconds with setPollRate (this also starts polling) and stop with stopPolling. When setting a poll rate, consider latency requirements, available bandwidth on the device, and available API calls on your

Heroku account.
    - There are two types of data you can poll: raw and aggregate/stats. Raw data is a replication of each user input into the database (e.g. each row in ParamList). Aggregate / stats is a set of statistical analysis of all audience data (e.g. mean, median, standard deviation, etc.). The stats only consider the most recent input from each user and discard data from users that have not been active in the last 60 seconds. For both, you specify which parameter id you wish to retrieve data for.
- Managing audience input tables:
    - startNewParamList creates a new table on the server for audience data with the current timestamp. This keeps tables from getting too long (which can affect performance on the server) and also keeps data from events in different tables to facilitate data analysis.
    - getCurParamListName ensures that all API calls get audience data from the most recently started paramList. This should always be called right after setting the API key.

APPENDIX C: About Python Analysis Scripts

We've created some simple utility scripts to help you export audience participation data from a massMobile event into a CSV file that you can import into a program like Excel for further review and analysis. Note that to use these scripts, you'll need to have access to the Heroku / mongoDB console, which generally means that you have to have your own (usually free) Heroku account (see Step 4 above).

How to use

- If you haven't downloaded MongoDB to your local computer, download it from http://mongodb.org. You can run the mongo runnable binary from within the your downloaded folder.
- Log into Heroku.com, select your app, and click on the MongoLab add-on to access the MongoLab console.
- Find the ParamList_XXX table of interest and export it as JSON using the mongodb command line tool you downloaded above. A straightforward ready to use command can be found in your mongolab main page, under the "Tools" section. Note you need to use the JSON format export command, like `mongoexport -h ds045xxx.mongolab.com:456xx -d heroku_app18175xxx -c <collection> -u <user> -p <password> -o <output file>`
- Use the reorganize.py script, located at ./dataSets folder, to preprocess the exported JSON file. (Type "python reorganize.py" into your command prompt.) Use your text editing software to open reorganize.py and manually set the input and output file names, then run the script.

- The preprocessed JSON file can then be analyzed. Please open the analyzeParamList.py in your text editor to manually set the location of the preprocessed JSON file (the one that was output by reorganize.py). If there is a python package you are missing (possibly the dateutil.parser module), please download and update your python to the latest 2.x version of your platform (https://www.python.org/download/). And be sure to set the correct project ID, parameter ID and parameter type in the following lines. You need to use an integer to represent the corresponding parameter type. Use the chart below or locate these values by manually inspecting the data in your mongoDB parameter list.

paramType number	parameter type
0	1D slider
1	2D slider
2	vote
3	text
4	draw
5	palette

The above are the works required before doing the real analysis.

Statistics Analysis Functions

The analysis script is used as you are writing a python script. The original contents in the analyzeParamList.py file is an example of how to use the provided functions. Normally you should not touch the function details in func.py. Note that, normally, all the *isTimeInInteger* parameter in the following functions should be set to *True.*

To get basic overall user activity stats:

funcs.getBasicParticipatingStats(data, projId, paramId, paramType)

Explanation: data is the retrieved JSON data as indicated in the analyzeParamList.py. This will print total user input counts, number of users, number of inputs per user, median of inputs, etc.

To get basic overall user activity stats in a time range:

funcs.getBasicParticipatingStatsInRange(data, projId, paramId, paramType, startTime, endTime)

Explanation: the same as the above function, distinct in that it looks it data with a specified time range.

Export a csv file with time sliced medians of parameter values:

funcs.paramValuesMedianAlongTime(projId, paramId, paramType, data, manualStartTime, manualEndTime, useManualTime, timeIncrement, isTimeInInteger)

Explanation: start/endTime are timestamps in long integer form representing the milliseconds since epoch time; the useManualTime field is a boolean value which determines whether start/endTime arguments will take effect. If True, the arguments will be used as start and end time as the range to be analyzed. If False, the script will automatically find the earliest time as the start time and the latest time as the end time to analyze. So be sure the data in your ParamList does not cover a super long time range, say several hours. Since the analysis will look at each time slice and advance the time to look at depending on the timeIncrement argument. If your time range is very large and the timeIncrement is a small step, the export data will be very large and takes a lot of time to process; the timeIncrement is in milliseconds and works as the time increment as indicated; the last TimeInInteger field tells the functions whether the start/endTime arguments are in integer representing milliseconds. Normally just keep this as True, since the timestamps used in the ParamLists should always use integers.

Export a csv file with time sliced averages of parameter values:

funcs.paramValuesAverageAlongTime(projId, paramId, paramType, data, manualStartTime, manualEndTime, useManualTime, timeDelta, isTimeInNumber)

Explanation: Similar to the time sliced median function, except this gets the average.

Export a csv file with time sliced standard deviations of parameter values:

paramValuesStandardDeviationAlongTime(projId, paramId, paramType, data, manualStartTime, manualEndTime, useManualTime, timeDelta, isTimeInNumber)

Explanation: Similar to the time sliced median function, except this gets the standard deviation.

Export a csv file with time sliced user activity statistics:

userActivityAlongTime(projId, paramId, paramType, data, manualStartTime, manualEndTime, useManualTime, timeDelta, isTimeInNumber)

Explanation: Get time based user input activity. It show the number of total inputs and the

number of total inputs from unique users. Arguments are similar to the above.

Export a csv file with time sliced user input raw data of a specific user:

userInputsAlongTime(projId, paramId, paramType, data, manualStartTime, manualEndTime, useManualTime, timeDelta, isTimeInNumber, userId)

Explanation: Get time based input data of a specific user. For example, for 2D slider parameter, it will collect the user's X and Y inputs at each window. Arguments are similar to the above, with an additional userId field containing the user ID string.