Turk Talk Instruction Notes

There are a few steps needed in setting things up before you can make use of our Turk Talk interface. But first of all, why would you use Turk Talk?

## Background:

Turk Talk has been designed for small group work, and in particular, how to improve the ability of OpenLabyrinth to handle free-text input in such situations. A common challenge noted by learning designers when working with virtual patient cases is that the users are always picking from a list of pre-defined options. This creates the problem that the user is largely cued about the possible responses – and yet there are many situations where the teacher or case author does not want to prompt the learner about what is the expected response.

The usual approach taken with this is to use a free-text input box where the user can type in their response. If you only have a limited number of participants, and you have the time to evaluate these responses after the session, then the teacher can simply evaluate each response given.

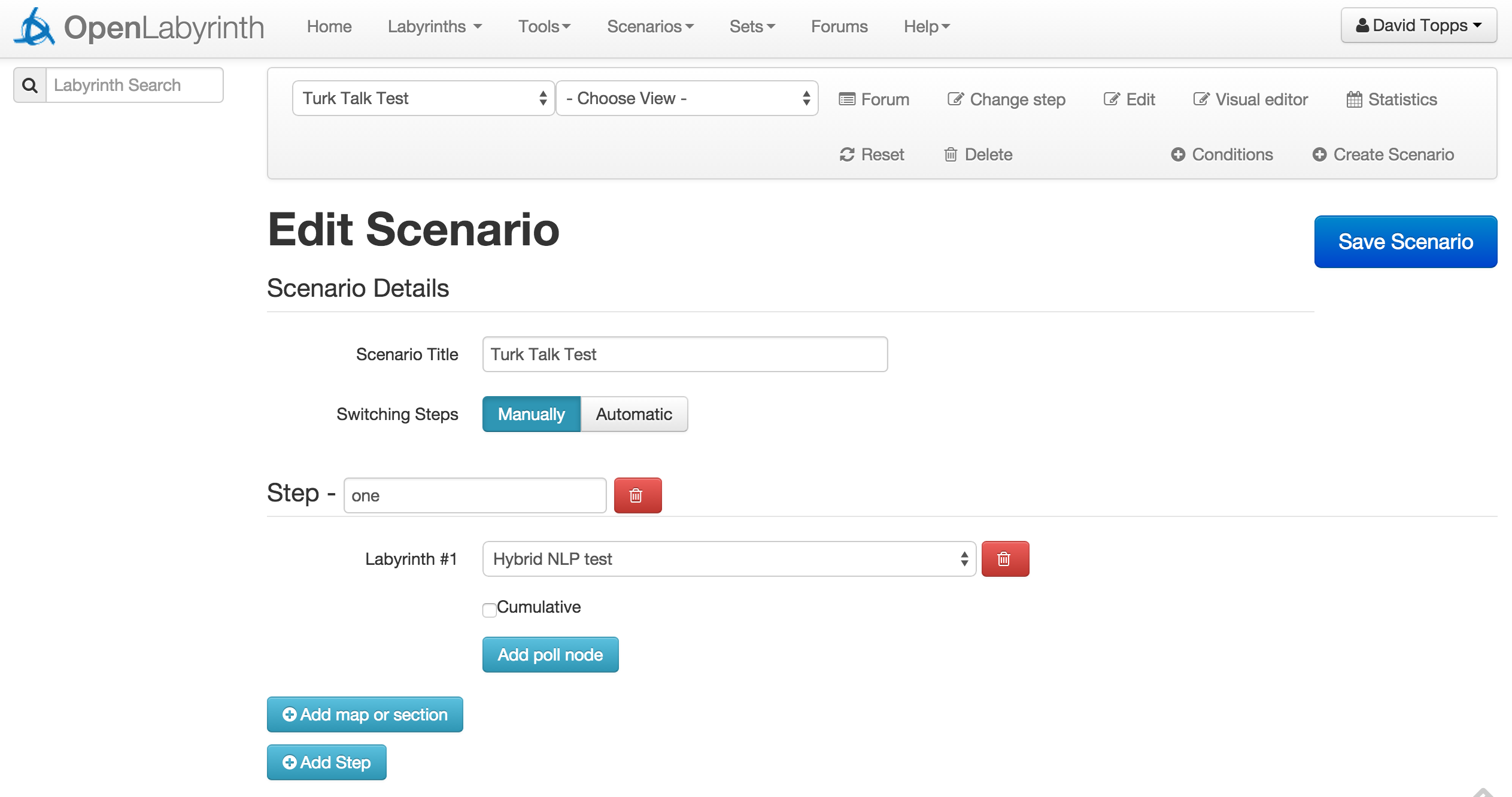
If an immediate response is needed then you can do some limited automatic processing of the free-text input by OpenLabyrinth. You can use the Rules or Question Rules for this. Consult those sections in the User Guide for more detail on how to use these. While the syntax allowed in these Rules is quite flexible, case authors will still have the problem of trying to imagine all the possible words, phrases and abbreviations that learners might employ in their response, e.g. for a chest pain case, you might need to allow for all of the following and more: “angina”, “angina pectoris”, “myocardial infarction”, “myocardial infarct”, “MI”, “M.I.”, “STEMI”, “myocardial ischemia”, “cardiac ischemia” – thinking of all those options is quite daunting. For some situations, the possible variations in phrasing will be very hard to cover. And as soon as you allow for more than 2-3 words, you are into natural language processing and this becomes very difficult indeed.

In Turk Talk, we have created the option for live intervention by an online teacher. The history behind the Mechanical Turk is interesting but the concept has come to mean using a human to simulate a computer. In this case, we are using a human-hybrid approach where the Turker or Scenario Director provides the natural language interpretation for short answers captured in the free text fields.

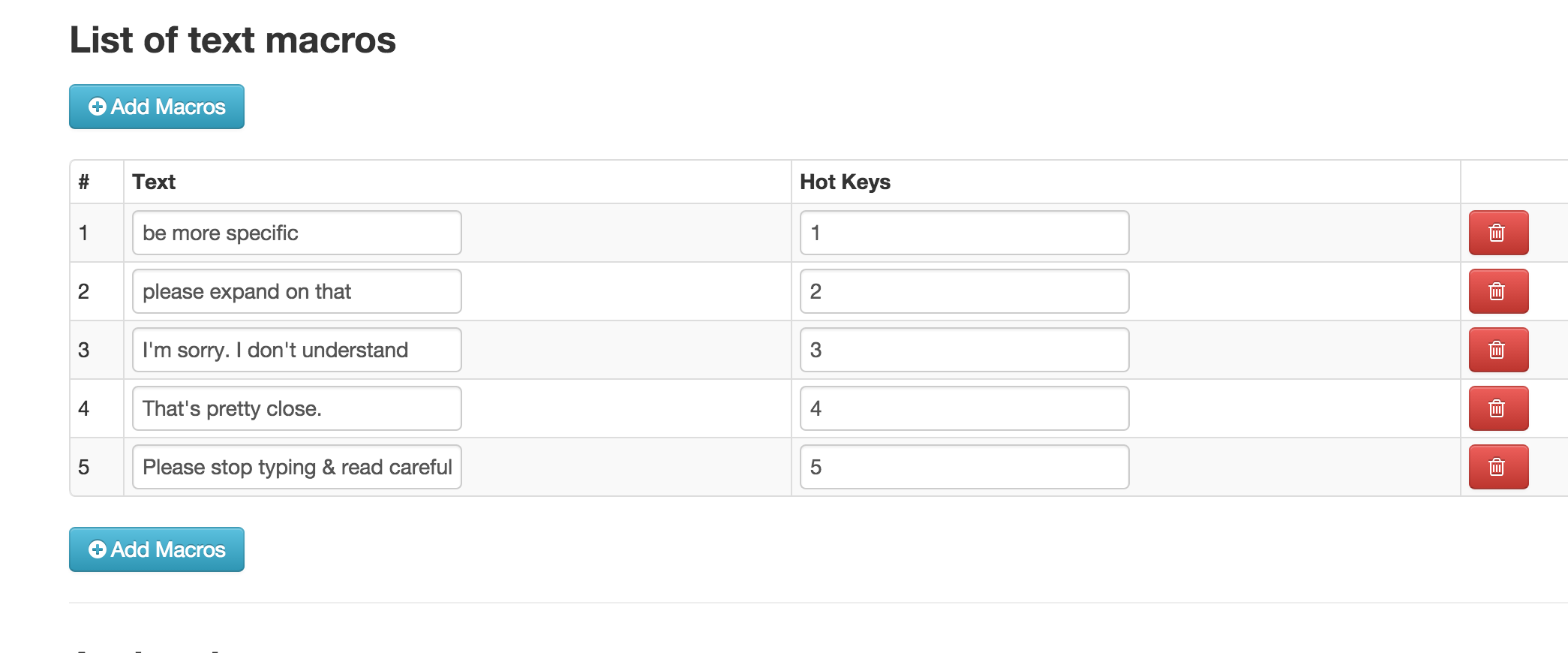
As mentioned, Turk Talk has been designed for small group work. Currently the Director screen is optimized for up to 8 concurrent learners. This may be modified in future. Because the amount of text to be interpreted is short and the variations limited, this approach may be scalable to a larger number of concurrent users.

## Turk Talk Setup:

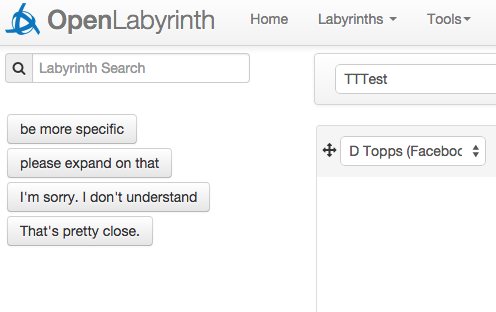
Turk Talk must be used in combination with OpenLabyrinth Scenarios. For more information on Scenarios, [see the online User Guide](http://demo.openlabyrinth.ca/documents/UserGuide.pdf). First set up your Scenario to include the case(s), the Steps and the Users you want to include.



Next add the Macros for the Turker to use while interacting with the learners. This step is new and specific to Turk Talk. Macros are simply some pre-defined text snippets that the Turker can insert into a chat thread, using a single keystroke. Each Scenario has its own macro set.

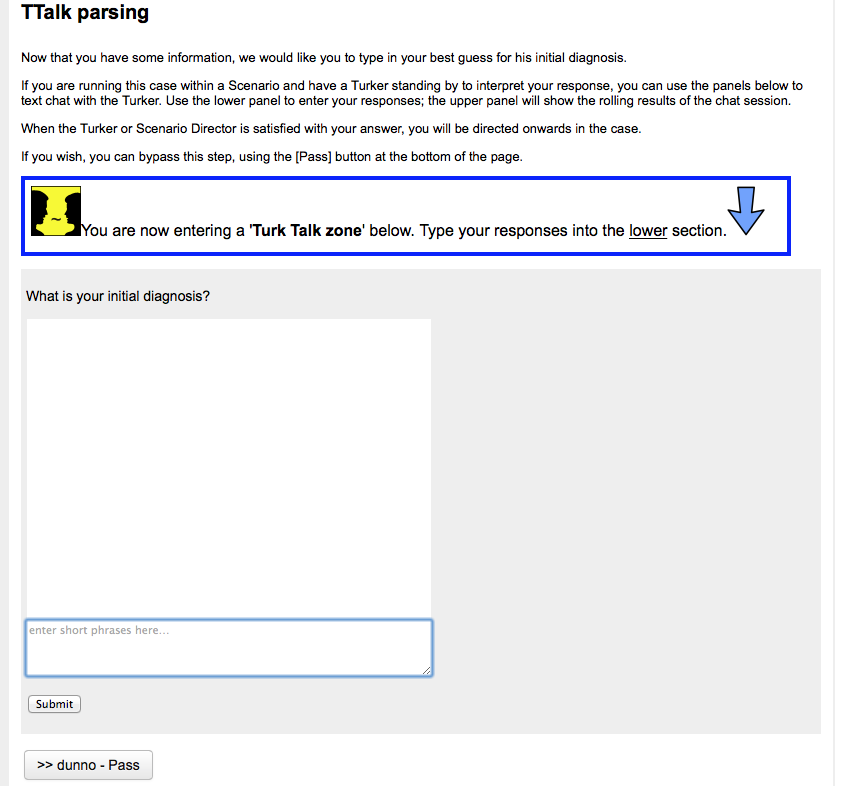


The macros are each assigned to a hotkey (Ctrl-1 to Ctrl-9 for Windows; Opt-1 to Opt-9 for Mac). When the Scenario Director or Turker is interacting with the group of Users, to insert the macro’s text, click on the chat column desired and then use the hotkey or click on the macro’s button in the top left of the chat view.



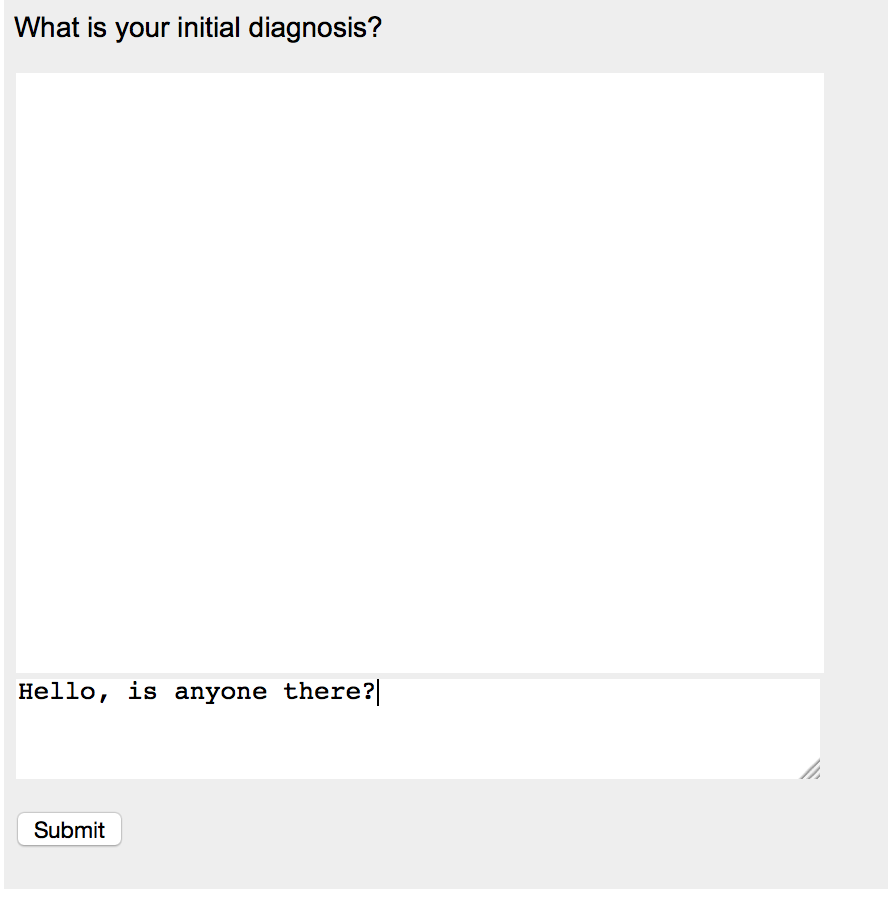
## Playing a Scenario using Turk Talk

From the User’s perspective, there is little difference in what they see. For most nodes in the labyrinth, they simply interact with OpenLabyrinth in the normal manner. When they arrive at a Node which allows interaction with the Turker, they will see a panel like this:



This user has reached the first Turk Talk node. Text is entered in the lower portion of the dialog box. Note the manually added highlighter box drawing attention to this. (We created this as an OpenLabyrinth Element, allowing it to be reused at will.)

The upper panel will hold the contents of the chat between the User and the Turker thus far. The User sends messages to the Turker using the lower section:



When the User presses the Enter key or clicks [Submit], the text is sent to the Turker and appears in the panel above. The Turker’s replies will also appear in the panel above. Note that this will only work if: (1) this labyrinth is being played within the Scenario; AND (2) the Turker is online. Because the Turker is handling up to 8 concurrent Users, there may be a short delay before the Turker responds. The User can send another message while waiting. Here we see the Turker’s response:

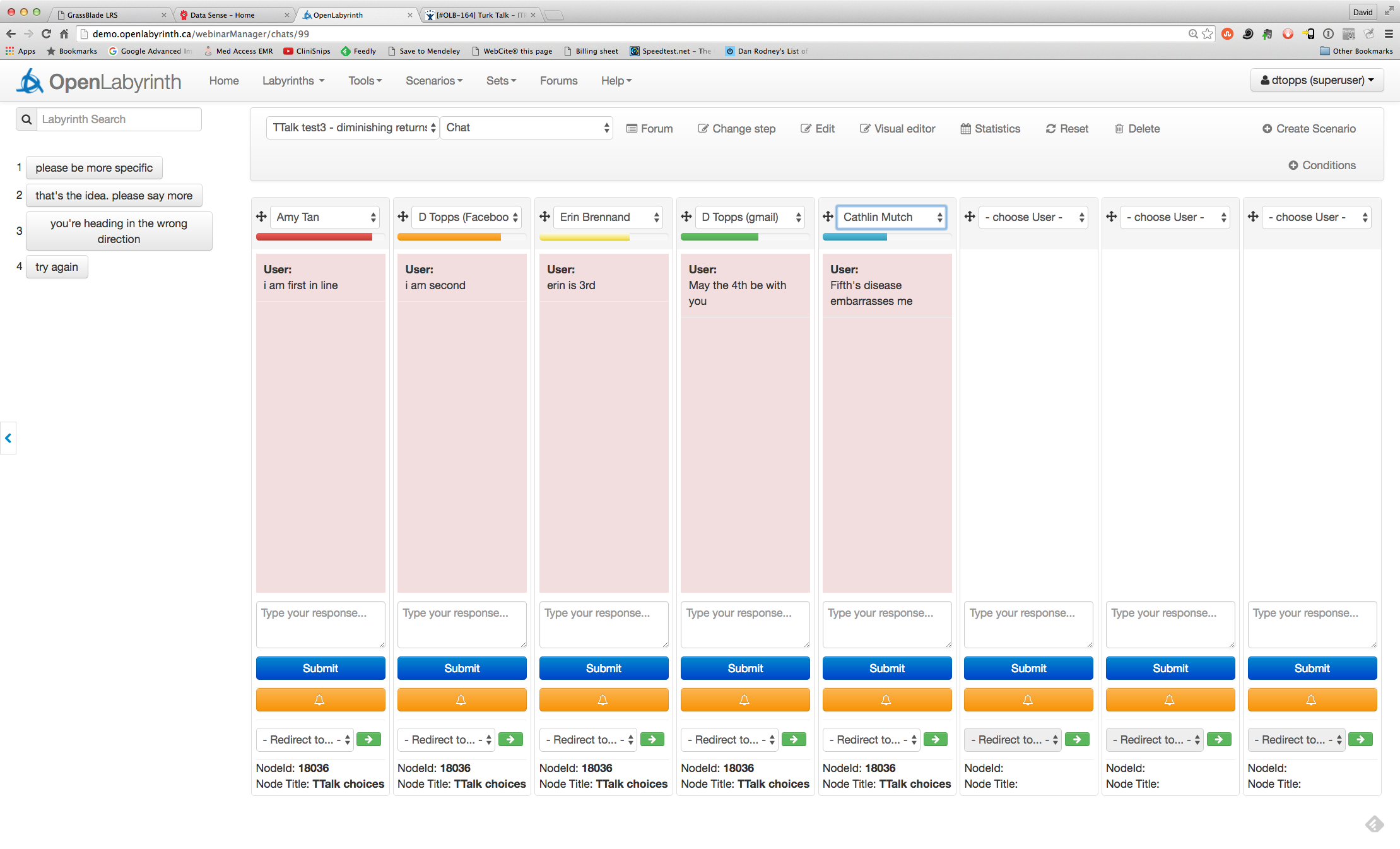


Once the User has satisfactorily answered the Turk Talk question that has been posed, the Turker will direct the User on to the appropriate pathway in the labyrinth and the User’s play will continue as usual.

## Directing a Turk Talk Scenario

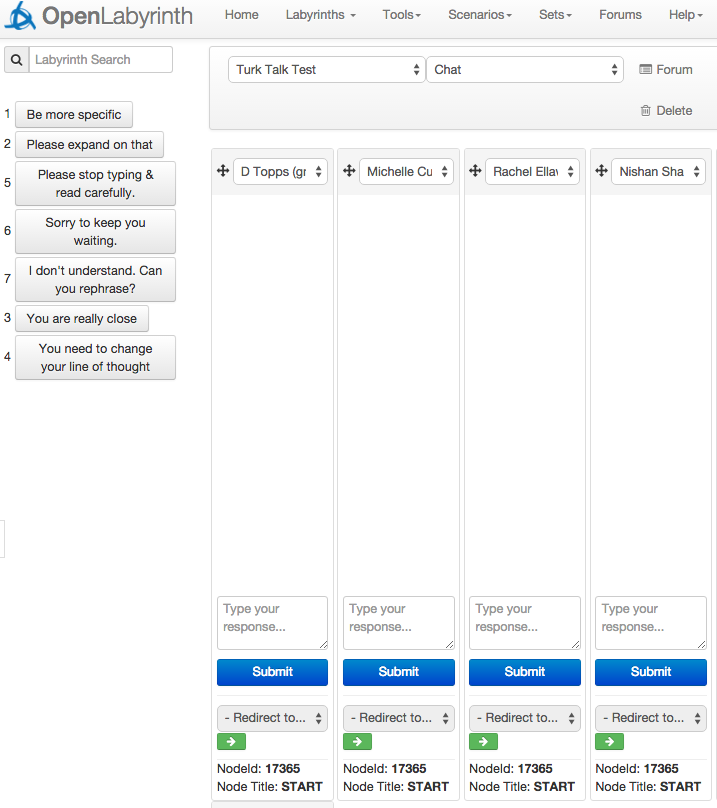
When a Scenario Director is running a Scenario that includes Turk Talk questions, they may need to be on their toes because interactions with Users will tend to occur in flurries of activity. Some practice will be needed for smooth operation.

Open the Scenario and change to the Chat view using the drop-down at the top of the page.



The view above shows the maximum 8 column view. It also shows five users waiting for a response from the Turker. This design is the latest iteration. You will also note that there is a Bell button for each column – this is for the Turker to nudge the remote user.

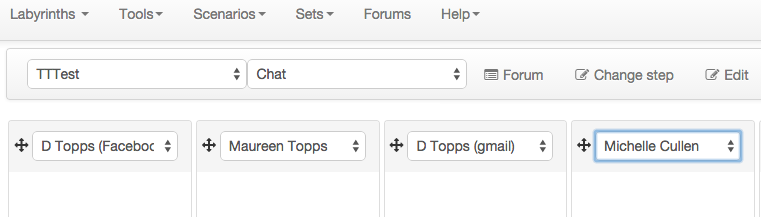
(In the screenshots that follow, some are from older designs where the [Submit] button was green, or the redirect arrow was slightly lower. We trust that readers will be able to adjust to these minor changes.)



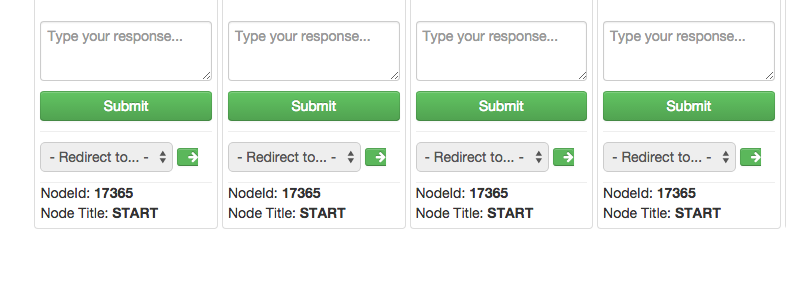
Scenario Director’s view when all Users are waiting on the START node:

Note up in the top left corner, the list of text macros that the Scenario Director or Turker can fire with one press of the hotkey or by clicking on the button.

Next, at the tops of the columns, you can see the current Users, each in their respective column. The Turker can assign any column they choose to the Users who have logged into this Scenario. The Turker can also drag active columns into any order from left to right, using the small crossed arrows. Make sure that each User has been assigned to a column. This is not done automatically.

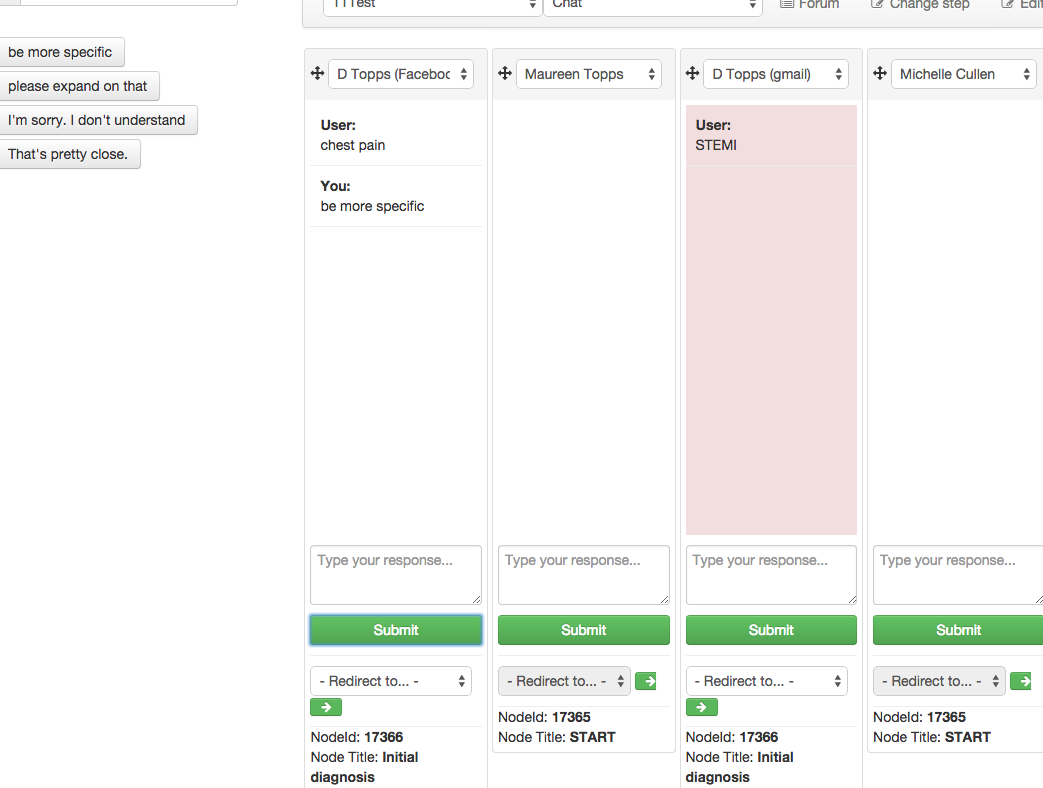


Down at the bottom of each column is the action area. Currently, this is showing that all 4 participants are waiting on the START page (node number 17365):

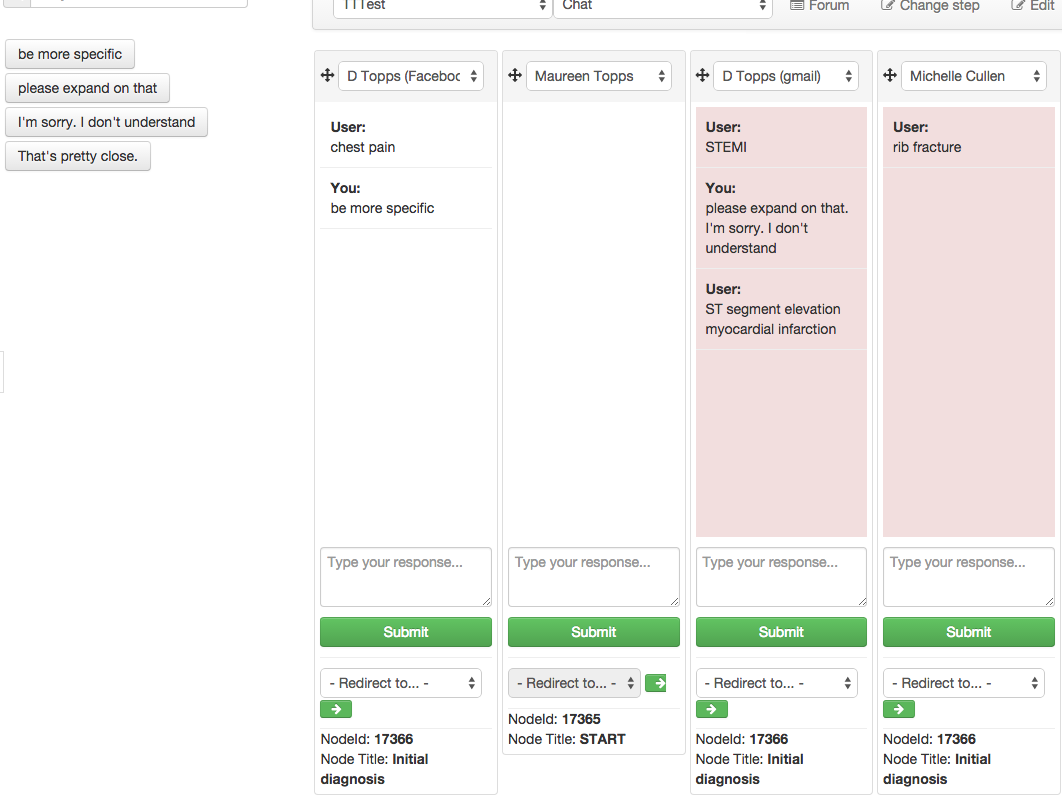


Also in this action area, is the [Submit] button for each column (which is now blue). Below this is the drop-down list of possible Nodes that the Turker can send the User on to, when they are satisfied with the response. Note that only those nodes that are linked from this Node in the overall labyrinth design are included here. When the User is not on a Turk Talk node, this redirection is not active and is greyed out, as above.

In the next figure, now that the Users have reached the first TTalk session, you can see that things are starting to happen:



In the above, the Users in the first and third column have reached the Initial Diagnosis node and have entered some text. In the first column, the Director or Turker has replied, asking the User to ‘be more specific’; in the third column, the column is pink because the User is awaiting a response from the Turker. Note also that the redirect list is no longer greyed out, allowing the Turker to pick from this list of destination Nodes.

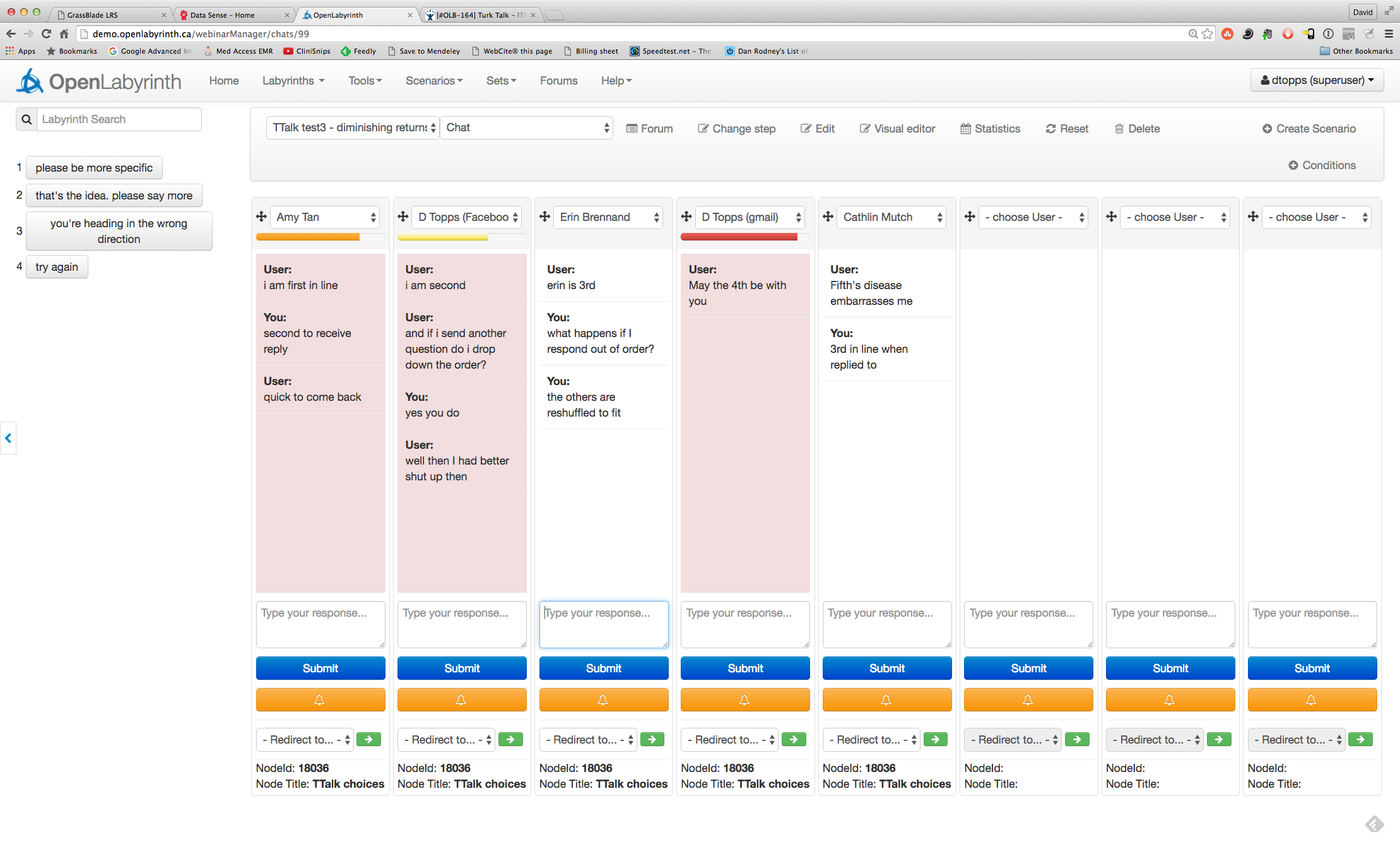


In the above image, the 4th column has joined the TTalk session and is awaiting a response; the 3rd column has received a response from the Director (this actually shows that you can concatenate two macros in the same response), and has added more information so is again awaiting a response from the Director.

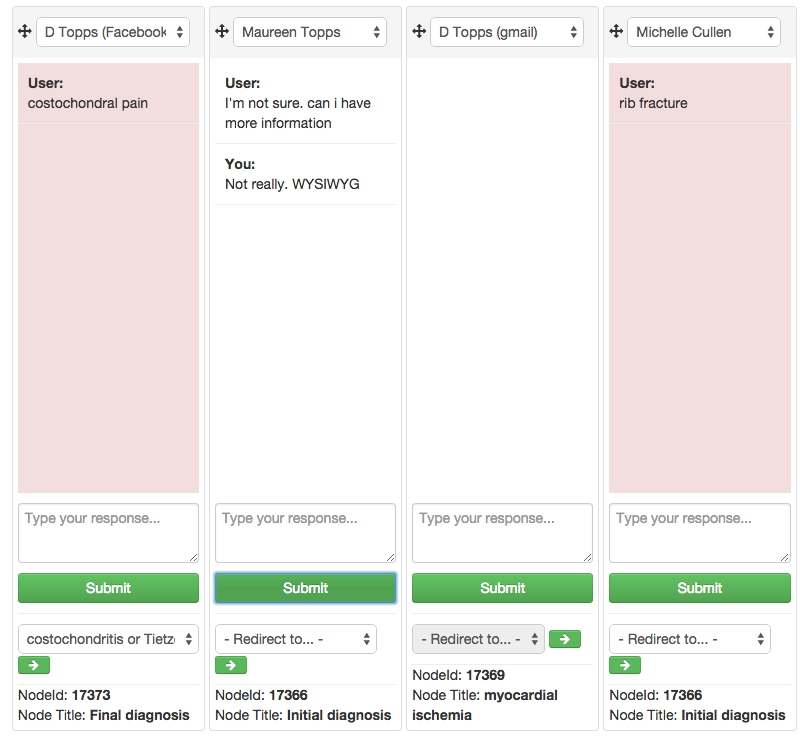
In the latest design, as shown on the first shot of the Turker screen above, there is now a color coding scheme to show which user has been waiting longest:

1. red
2. orange
3. yellow
4. green
5. blue

All beyond 4th in line are colored blue. We have found it is rare to exceed this. These colors remain accurate even if the Turker chooses to respond in a different order – they readjust according to the new order of priority.



The waiting order has now changed after a few responses. The red bar now shows that column 4 has been waiting longest.



In the above image, the User in the 1st column has moved on in the case and has reached the Final Diagnosis stage and is awaiting a response from the Director. The Director has selected ‘costochondritis…’ as the satisfactory equivalent response and is about to send User 1 on to that node. The Turker will click on the green arrow to redirect the User.

User 2 has reached the Initial Diagnosis TTalk stage and the Director has replied. User 3 has been moved on to the ‘myocardial ischemia’ node which was the closest option for their response. (Their chat column has been cleared to minimize distraction). User 4 is still awaiting a response from the Director to their suggestion of ‘rib fracture’ – this is now a bit overdue.

As you can see, it may be quite challenging for the Turker to stay on top of things and keep all 8 Users happy. Good labyrinth and Scenario design will help.

## Labyrinth design for Turk Talk:

This section has been optimized as we gained more experience with this approach, and yet, we welcome further input from others who are experimenting with Turk Talk. There are a few things to consider in the design of your cases:

1. Do not use free-text input for everything. While there is a strong history of this approach in Interactive Fiction engines, our research has generally found that strong use of free text input is not efficient as a means of case navigation and clinical reasoning.
2. Do not use free text when it is perfectly reasonable to present a set of choices as either Nodes or Questions. Most of the time, there is no reason why Users should not be able to pick from pre-assigned choices.
3. There are other ways of making the correct choice less obvious than using free text. You can present a larger number of options (it has been suggested that providing 12 – 20 choices is a more optimum range than the usual 4 – 6). Or you can use approaches that avoid the standard single-best-answer method: script concordance, situational judgment and key feature questions are some possibilities.
4. However, it is useful to use free text input when you want to completely avoid cueing the learner about possible responses. But if the range of responses is small and the variations on phrasing are limited, you may well be able to get OpenLabyrinth to handle the parsing of the free text entered by using Rules and Question Rules. See [User Guide for more on this](http://demo.openlabyrinth.ca/documents/UserGuide.pdf).
5. In our work on learning designs within OpenLabyrinth, we have typically found that you only need to use the Turk Talk approach in a labyrinth for about 2 – 3 questions out of 20 – 30. (i.e. about 10%).
6. Because you do not want all your users to be bombarding the Director or Turker with questions and chat right away in the labyrinth, we suggest that you place half a dozen questions before your first Turk Talk question so that the User demands are slightly staggered in timing.
7. Remember that if your Turker has more complex decisions to make regarding user input, such as whether the response is good enough, or possibly that the Turker has to spend more time prompting learners for a good answer, then the number of users or responses that they can handle will drop accordingly. Consider this in your case design.
8. If you want your Users to be comfortable with the Turk Talk approach, you could consider giving them practice time in chatting with the Turker at the very start of the Scenario. However, we would urge caution in this because the primary function for Turk Talk is rapid yet flexible interpretation of critical responses, and not as a help channel for those who are confused.
9. There is a new Question type that designed for Turk Talk. Similar to the multi-line text box, it provides a chat history of what has been typed by the User and the Turker in that particular mini-session. Edit and insert this Question type in the same manner as you would for other Questions.
10. For some cases, you will probably find that you need to provide your Turkers with a script, outlining the case, and what are acceptable responses at each Turk Talk node. We provide an [example script](http://openlabyrinth.ca/turker-script-for-scenario-95/) here.

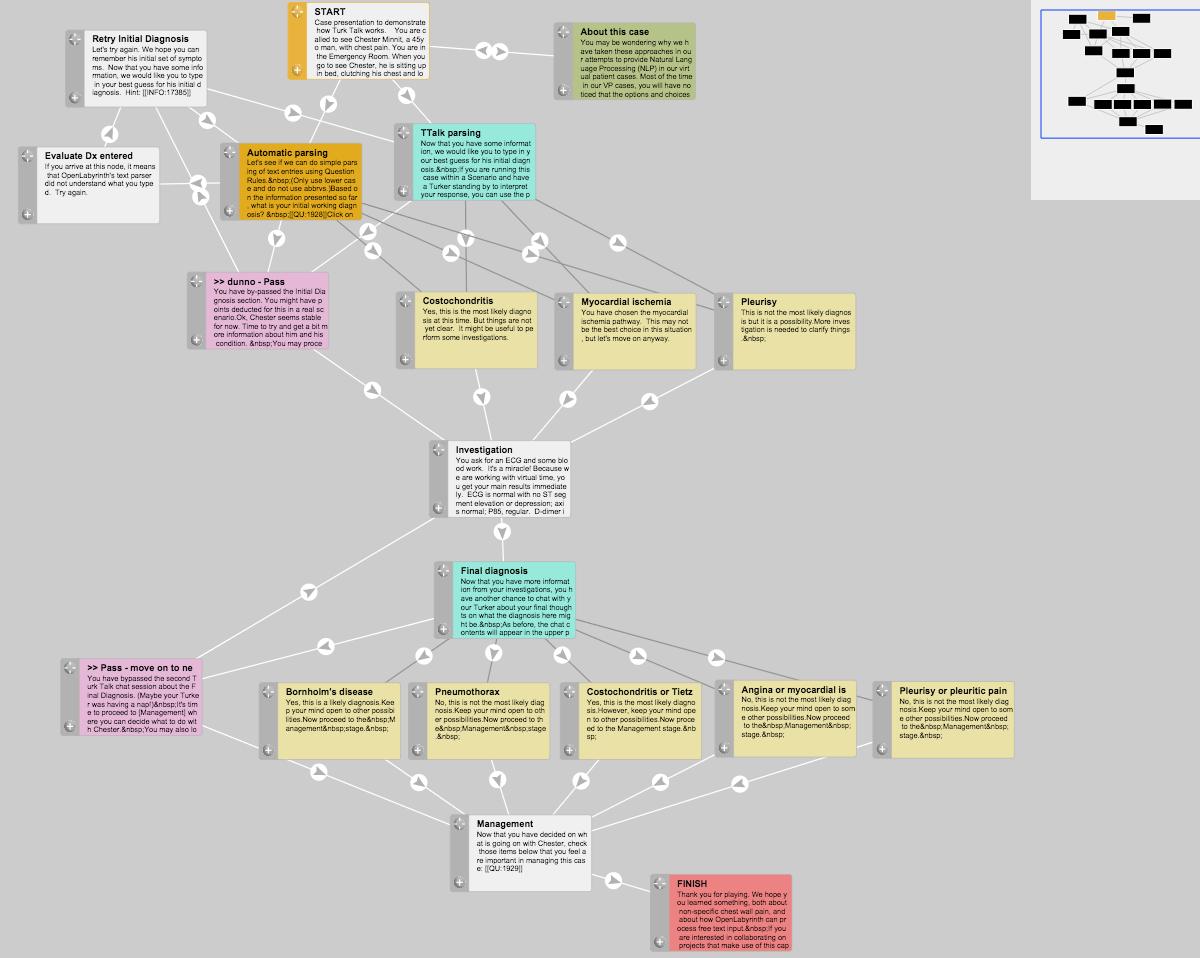
Suggestions for possible starting Macros:

1. Try again
2. Please tell me more
3. You are close
4. You are off track
5. I don’t understand
6. Read the question carefully
7. Try phrasing a different way
8. Good enough. I am sending you onwards
9. You seem stuck. I am sending you on

Obviously you can edit these to suit the context of each Scenario.

## Writing a Turk Talk case

As with other case designs, you will still generally use the Visual Editor to write a Turk Talk case. Here is the Visual Editor concept map for <http://demo.openlabyrinth.ca/labyrinthManager/global/606>, our first OpenLabyrinth case design, using the Turk Talk approach.



In this concept map, extracted from our first Turk Talk case design, the flow is from top downwards. Each Node contains a single page within the case or labyrinth, linked according to the arrowed lines. The dark grey links signify hidden paths that are not immediately apparent to the learner as a choice but that the facilitator can direct them along. To make a Link hidden, click on it and change the attribute.

For illustration purposes only, we have manually colored the nodes. The aquamarine nodes are Turk Talk nodes where the learner interacts with the facilitator. The light yellow nodes are the possible destination points that the facilitator can send the learner on to. The lilac nodes are by-pass nodes where the user can go if they give up waiting. The ochre node was a node with automatic text parsing, for comparison purposes.