Backend API endpoints created:

Landing page: (‘/’), METHOD: GET, POST
Redirects to /auth
Sends a 302

GET /auth

Random userID is generated
Random secret is generated
User object is added to db
Sends a 200

POST /select, with a body containing the _id/secret

Authorizes _id/secret
queueSelection.html (really the html for the landing page) is rendered
Sends a 200

At this stage, a client has gone onto the website and the website has automatically generated a userID and a secret for that client. This user object is added into the database, with:

queueType set to 0
timestamp set to an irrelevant time

When the client clicks on Talk, Vent, or Listen:
POST /queue, with a body containing _id/secret

Authorizes _id/secret
queueType and timestamp are updated

We have 2 options for how we want to implement rooms:

1) Via a new endpoint
2) Via an event that is triggered by the js script

Match-Making

The match-making service will have to find 2 users from the database and put them in a room. How this will work is this:

Rooms are kind of abstract in Socket.IO, where Users still send messages as usual. The only difference is that messages are only sent to other users that have the identical room field. So what we can do is, instead of having a system where users are put in pre-existing rooms, we can do this:

We’ll have a separate open room list or open room database of some sort

When a user is added onto the queue, check if the open room db has any rooms

If there is a room (meaning there is a room with one user waiting in it), join that room and then remove that room from the database of open rooms. Then render the chat.html for both users.

If there isn't a room, take the user's id. This will be used as the roomID. Have the user join a room, where the room ID is the user's id (this essentially creates a new room with a unique identifier). This roomID is added to the database. Wait for someone else to be on the queue, and then that other user should be added to the room. Then remove the room from the database and render the chat.html

This way we can have each user either create a new room with a unique identifier or join a room. There won't be some pre-existing list of rooms that users join.

It is worth noting that in the handleMessage() method, in send() there is a room field. This specifies which room the current user sends messages to.