Introduction and Message Structure:

The iSENSE network protocol is built on messages. Messages consist of a message code along with optional parameters. Java had built-in classes for dealing with streams of typed data, so it makes sense to use them for this project. Specifically, the main two classes we will use are called DataOutputStream and DataInputStream. Using these two classes, a Java developer can use simple code like outputStream.writeInt(500) to send an integer over a stream. The stream can be pointing to a socket, which is helpful in our case.

A network message has the following structure:

<table>
<thead>
<tr>
<th>FIELD:</th>
<th>TYPE:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESSAGE CODE</td>
<td>Integer (4 bytes)</td>
</tr>
<tr>
<td>PARAMETER COUNT</td>
<td>Integer (4 bytes)</td>
</tr>
<tr>
<td>PARAMETER TYPE</td>
<td>Integer (4 bytes)</td>
</tr>
<tr>
<td>PARAMETER DATA</td>
<td>Integer, Float, Double, Boolean, Short, Character, Byte, Long, UTF-encoded String</td>
</tr>
</tbody>
</table>

Although this implementation uses Java's DataOutputStream and DataInputStream classes, other languages could interoperate with it fairly easily. Java does make it very easy though.

Responses:

The protocol is based on a simple request-response structure. The client requests something using a message, and the server responds with another message. Everything from user authentication to data manipulation is performed using messages and responses. A response can either be of type MSG_SUCCESS or MSG_FAILURE.

A MSG_FAILURE message, as its name implies, indicates there was a problem with the client's request. An error message (String) can be attached to a MSG_FAILURE message. A MSG_SUCCESS message indicates the request was successful, and the attached parameters vary depending on which message the client has sent.

The responses take the following form:

200. MSG_SUCCESS <varies depending on client's request>
Indicates the client's request succeeded. The message's parameters vary depending on the request the client made. See below for more information on MSG_SUCCESS parameter formats.

201. MSG_FAILURE <String message>
Indicates the client's request has failed. The error message (String) is optional in all cases.

Data Types:
Data type identifiers are represented using simple Integer values. The values used in this protocol are as follows:
<table>
<thead>
<tr>
<th>Type</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>50</td>
</tr>
<tr>
<td>Double</td>
<td>51</td>
</tr>
<tr>
<td>Float</td>
<td>52</td>
</tr>
<tr>
<td>Short</td>
<td>53</td>
</tr>
<tr>
<td>Character</td>
<td>54</td>
</tr>
<tr>
<td>Byte</td>
<td>55</td>
</tr>
<tr>
<td>String</td>
<td>56</td>
</tr>
<tr>
<td>Boolean</td>
<td>57</td>
</tr>
<tr>
<td>Long</td>
<td>58</td>
</tr>
</tbody>
</table>

See the list of client requests for more information on how these data type identifiers are used.

**Client Requests:**
The client can send any of the following requests. The number preceding each message is the actual integer value used to represent the message.

100. MSG_AUTHENTICATE <String username> <String password>
Authenticates the user against the Moodle user database. If the user credentials are correct, a MSG_SUCCESS response is sent back to the client with an optional String message, and the session can continue normally. If the credentials are rejected, a MSG_FAILURE response is sent to the client with an optional error message (String), and the connection is terminated by the server.

101. MSG_CREATE_EXPERIMENT <Integer experiment_name>
Creates a new experiment, with a given name, in the database. If the experiment is successfully created, a MSG_SUCCESS response is sent to the client with an experiment identifier (Integer). If the experiment could not be created for some reason, a MSG_FAILURE response is sent to the client with an optional error message (String).

102. MSG_GET_EXPERIMENT_LIST
Retrieves a list of all experiments owned by the user. If the list is retrieved successfully, a MSG_SUCCESS response is sent back to the client. The MSG_SUCCESS response is required to have, as parameters, all the experiment identifiers (Integers), or zero parameters if no experiments exist. If the list of experiments cannot be retrieved for any reason, a MSG_FAILURE response is sent to the client with an optional error message (String).

103. MSG_DELETE_EXPERIMENT <Integer experiment_identifier>
Deletes a given experiment from the database. If the experiment is successfully deleted, a MSG_SUCCESS response is sent to the client. If the experiment cannot be deleted for any reason, a MSG_FAILURE response is sent to the client with an optional error message (String).

104. MSG_ADD_DATA_FIELD <Integer experiment> <String name> <Integer type>
Adds a data field to a given experiment, giving the data field a given name and data type. If the data field is added to the experiment, a MSG_SUCCESS response is sent to the client with a data field identifier (Integer). If the field cannot be added to the experiment for any reason, a MSG_FAILURE response is sent to the client with an optional error message (String).
105. **MSG_GET_DATA_FIELD_LIST**<Integer experiment>
Retrieves a list of data field identifiers for the given experiment. If the list is successfully retrieved, a MSG_SUCCESS response is sent to the client with a list of data field identifiers (Integers) attached. If the data field list cannot be retrieved for any reason, a MSG_FAILURE is sent to the client with an optional error message (String).

106. **MSG_GET_DATA_FIELD_TYPE**<Integer experiment> <Integer field>
Retrieves the data type associated with a given data field. If the data type is successfully retrieved, a MSG_SUCCESS response is sent to the client with a data type (Integer). If the data field type cannot be retrieved for any reason, a MSG_FAILURE response is sent to the client along with an optional error message (String).

107. **MSG_DELETE_DATA_FIELD**<Integer experiment> <Integer field>
Deletes a data field and all associated data from the database. If the data field is successfully deleted, a MSG_SUCCESS response is sent to the client. If the data field cannot be deleted, a MSG_FAILURE response is sent to the client with an optional error message (String).

108. **MSG_STORE_DATA**<Integer experiment> <Integer field> <Integer type> <Integer count> <Variable data> <Timestamp time> ...
Stores one or more pieces of data in the database. The DATA field format varies depending on the type of data being stored. If the data is stored successfully, a MSG_SUCCESS response is sent to the client. If the data cannot be stored for any reason, a MSG_FAILURE response is sent to the client with an optional error message (String).

109. **MSG_READ_DATA**<Integer experiment> <Integer field>
Reads all data points from the given field. If the data is successfully retrieved, a MSG_SUCCESS response is sent to the client with a data point count (Integer). For each data point, there are two more parameters added to the response. The first in each pair is the data (in String format; conversions to the correct type can be done by the clients). The second parameter is the timestamp value associated with the data point. If the data cannot be read for any reason, a MSG_FAILURE response is sent to the client with an optional error message (String).

**Typical Client-Server Session:**

The typical client-server session begins with the client connecting and authenticating using the MSG_AUTHENTICATE message. If it succeeds, the connection is registered server-side as being tied to that account. Next the client executes any of the experiment, data field, and data store/read commands. It sends a message such as MSG_CREATE_EXPERIMENT, receives a response such as MSG_SUCCESS, and continues. When the client is done, it sends a MSG_GOODBYE message to the server. There is no response associated with this message, so the client may close the socket at immediately. The server should close its socket upon receipt of the MSG_GOODBYE message.

Here is the overview of the above session:

**Client:** MSG_AUTHENTICATE myusername mypassword
**Server:** MSG_SUCCESS “Welcome to my server”
**Client:** MSG_CREATE_EXPERIMENT ”Temperature vs. Humidity”
**Server:** MSG_SUCCESS 1
**Client:** MSG_GOODBYE
(client immediate closes socket)
(server closes socket upon receipt of MSG_GOODBYE)