To: Overlord Founder of our Precious Startup
From: Ryan Louie and Kapil Garg
Date: 10/01/2018
Title: Programming Language for the Next Unicorn

We will be using **Node.js** for this project. Both Ryan and Kapil have significant experience with the language, and Node.js meets the project requirements, as detailed below.

1. **Compilation to Unix executables that run on the school's lab machines and your laptops**
   Node JavaScript code can be compiled into a Unix executable using the nexe package (https://github.com/nexe/nexe).

2. **Support for UNIX-style STDIN, standard I/O and TCP/IP sockets**
   Node supports STDIN/STDIN out through readline (https://nodejs.org/api/readline.html) and console (https://nodejs.org/api/console.html) respectively. For streaming input, fs can be used (https://nodejs.org/api/fs.html#fs_class_fs_writestream). Finally, for TCP socket connections, we can use net (https://nodejs.org/api/net.html).

3. **Modular programming**
   Functors are supported natively (https://hackernoon.com/functors-in-javascript-20a647b8f39f). Newer versions of Node and JavaScript add "classes" that allow us to replicate object-oriented programming practices (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes). Node also has access to a vast package archive through the Node Package Manager (NPM, https://www.npmjs.com/).

4. **Reading and writing JSON**
   Fs can be used to both read and write JSON (https://nodejs.org/api/fs.html). Further, JavaScript has built-in functionality for parsing a string into JSON (https://www.w3schools.com/Js/js_json_parse.asp), and for outputting JSON as a string (https://www.w3schools.com/Js/js_json_stringify.asp).

5. **Loading code dynamically**
   Node allows code to be dynamically imported in as needed, both at the module level and function level (https://nodejs.org/api/modules.html).

6. **Automatic unit testing and test coverage**
   We can test our code using the Mocha testing framework (https://mochajs.org/), the Chai assertion library (https://www.chaijs.com/), and the Istanbul code coverage framework (https://istanbul.js.org/). Further, to create testing harnesses, we can use child process (https://nodejs.org/api/child_process.html) to execute other executables against our test cases.

7. **An IDE with support for exploratory programming.**
   We will use Webstorm (https://www.jetbrains.com/webstorm/) as our IDE since it provides good debugging tools and allows us to run code directly from the IDE.

From the above, we believe that we will successfully be able to complete the project using Node.js.