Building a scouting application with *electron*

2056 Ways to Inspire Conference – 2018

Ethan Elliott

A Bit of History

- Almost always paper-based scouting with Excel Spreadsheet
- Have only ever been paperless in 2011
- 2011 setup required a lot of hardware

Why stop paper-based now?

- Based on the required data for 2018 game
- Needed a way to better track time between events, to calculate cycle times
- Students aren’t great at counting time and watching the game
- By offloading a lot of the work for the student, we can have them collect more data
- Preform data sanitizing while the student is entering data

If you want to know more about the strategy…

- CHECK OUT THE PRESENTATION AT THIS TIME IN THIS ROOM WITH THESE GREAT PEOPLE

Where to start?

- Given the problem of building an entire scouting system across a full stack, where do you start?
- Which tools should we use?
- What is the available hardware?
- What limitations should we take into account?
- How do we aim for the highest operational uptime?

What does it need to do?

- Needs to have three main components: Server, Admin control, Student control
- Server needs to control all communication between everyone
- Admin control must be able to receive data from the server, and control the student application
- Student control must be able to allow student to enter data, and send that data to the server
- Must export data to excel format, for better analysis
- [illegible]

Restrictions

- Everyone needs a laptop – not everyone has the same laptop (must be cross-platform)
- Server must run on a laptop
- Power might not be available in the stands
- Cannot use Wi-Fi
- Available space is very limited in the stands

My Choices

- Electron – Admin, Student control interfaces
- NodeJS – Cross-platform client-side JavaScript execution
- ExpressJS – Simplified routing for building an API
- SocketIO – UDP socket connection system
- jQuery – Simplified client-side scripting
- DiskDB – Simple JSON file-based database system

NodeJS

- It’s a JavaScript runtime based on the Chrome V8 JavaScript engine
- Allows you to write JavaScript code to run locally
- Node is designed to build scalable network applications
- Perfect for a cross-platform server!

Electron

- Electron is an open-source framework developed and maintained by GitHub.
- Allows for the development of desktop GUI applications
- Uses NodeJS for the backend, and Chromium for the front-end rendering
- Build once, run anywhere (Cross platform!)
- Don't have to worry about polyfills, since everyone has the same engine

ExpressJS

- Minimalist web framework for NodeJS
- Provides a robust set of features for web and mobile applications
- Easiest way to build a simple API with NodeJS
- Simple methods for building application endpoints for a web API

DiskDB

- A Lightweight NOSQL-type disk based JSON Database with a MongoDB like API for NodeJS.
- All operations are simple look-ups on a JSON file, Handles all file system interactions for you
- Can connect to multiple dbs concurrently
- Reading, Writing, Updating, Deleting, Counting…
- No SQL server setup required!

SocketIO

- *Socket.IO enables real-time, bidirectional and event-based communication.*
- *It works on every platform, browser or device, focusing equally on reliability and speed.*

- Controls a real-time UDP connection between clients and the sever

- Socket traffic is handled by SocketIO, all you need to do are add events and responses

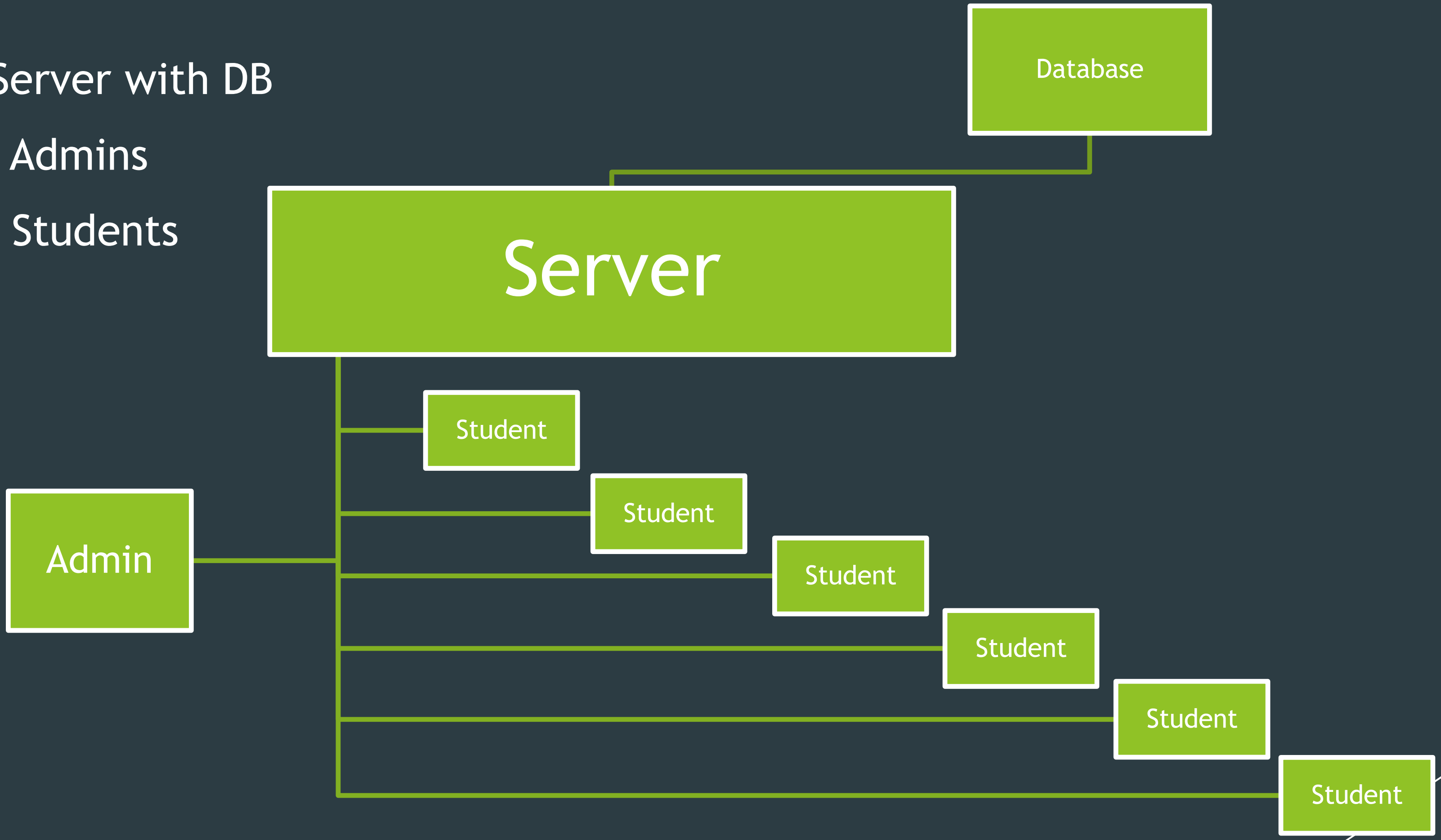
- Can control multiple connections simultaneously, and can separate groups of connections into *‘rooms’*

jQuery, SCSS, Jade

- jQuery is a library for simplified front-end scripting, building interfaces, and the like
- SCSS is pre-compiled CSS with variables and automatic polyfill
- Jade is pre-compiled HTML with control structures like loops, and if statements
- Together they offer a simplified way to build an advanced GUI

The System Overview

- 1 Server with DB
- 1+ Admins
- 6+ Students



Building Simple Interfaces

- Simple interfaces are required to simplify the process for the students
- The interface should be simple enough to become second nature while the student is scouting
- Colour coding and grouping of controls helps to offload thought from the student while they are trying to scout

How the server works

- The server is responsible for all inter-network communications, as well as storing and processing the data
- The server broadcasts itself on its network with a multicast address, so that other computers on the network can locate the server by themselves
- All events are controlled through SocketIO, causing the server to take certain actions, and respond to the request appropriately

How the Student application works

- Student application is the main data-entry interface
- The student application connects to the server through the multicast address, and establishes a socket connection
- The application will then respond to events, and allow the scout to enter data during the match

How the Admin application works

- The admin application is the main control center of the system. From here you can start/stop a match, and control where the data is going
- The admin application connects to the server through the multicast address
- There can be a theoretical infinite number of admins connected to the server

Gambling for in-between matches

- An informed scouting team, is an awesome scouting team
- Making bets on the successes of other robots requires an active knowledge of the performance of the robots
- Can only gain that knowledge by paying attention to the matches!
- This knowledge is very important for scouting meetings

Let’s run an example match!

Questions