Final Project

Let's run the final project!

WE'LL COVER THE FOLLOWING

In the previous lessons, you saw how to perform various operations. We inserted, read, updated and deleted users from our database, separately.

Now, let's combine all the codes from previous lessons and run the final project to see the output.

Project Implementation

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('mongodbnode:server');
var http = require('http');
var mongoose = require('mongoose');

var mongoDB = 'mongodb://127.0.0.1/blog';
mongoose.connect(mongoDB, {
  useMongoClient: true
});

//Get the default connection
var db = mongoose.connection;

//Bind connection to error event (to get notification of connection errors)
db.on('error', console.error.bind(console, 'MongoDB connection error:'));

/**
 * Get port from environment and store in Express.
 */
```

```
var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  var port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}

/**
 * Event listener for HTTP server "error" event.
 */

function onError(error) {
  if (error.syscall !== 'listen') {
    throw error;
  }

  var bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  // handle specific listen errors with friendly messages
  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
      process.exit(1);
      break;
    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;
    default:
      throw error;
```

```
  }
}

/**
 * Event listener for HTTP server "listening" event.
 */

function onListening() {
  var addr = server.address();
  var bind = typeof addr === 'string'
    ? 'pipe ' + addr
    : 'port ' + addr.port;
  debug('Listening on ' + bind);
}
```

Conclusion

The main purpose of this chapter is to show developers how to use MongoDB in a Node.js environment, but it should also serve as a small guide for starting with MEAN stack. After reading this chapter and analyzing the code, we hope that readers will be able to create simple web applications and continue learning about each technology inside the MEAN stack.

Before learning about Serverless & MongoDB Atlas, in the next chapter, let’s take a quick quiz.