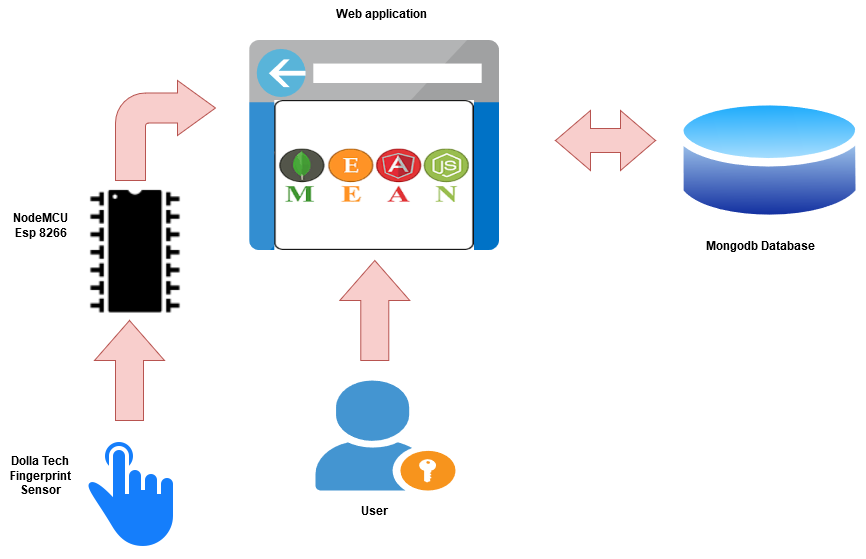


**Employee Management System**



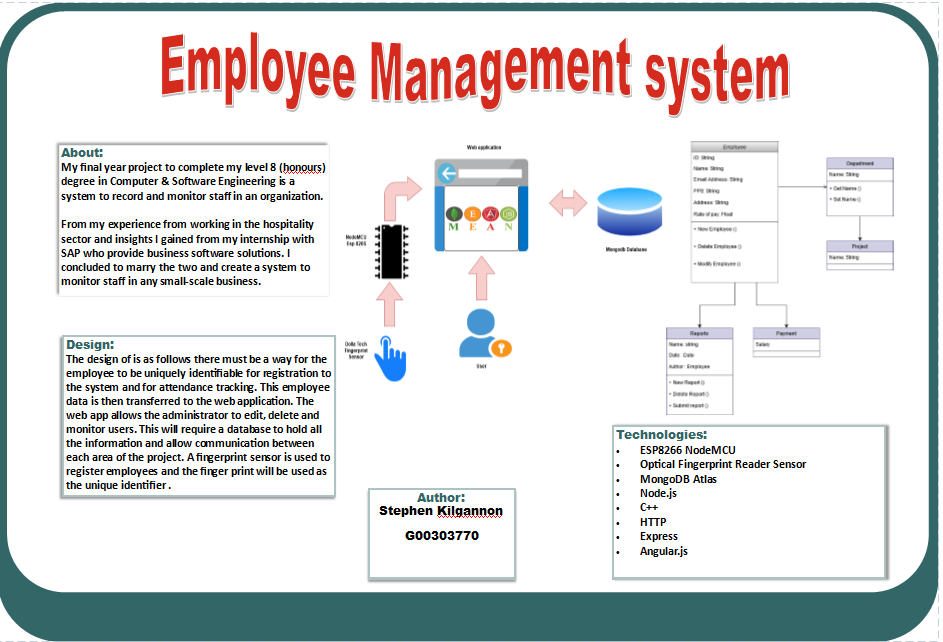
Stephen Kilgannon

**G00303770**

Bachelor of Software & Electronic Engineering

Galway-Mayo Institute of Technology

2019/2020



**Declaration**

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Software & Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acknowledgements**

I would like to thank to thank all the lectures and staff at GMIT whom have helped me throughout this year. Paul Lennon my supervisor whom gave me the freedom to work as I seen fit and was always ensuring that work was being carried out. Adafruit for there extensive work and libraries that they provide. Adam Bretz and Colin J.Irig who’s boked provided me the guidance to lean and understand working with the MEAN stack.

**Table of Contents**

[1 Summary 7](#_Toc40707176)

[2 Introduction 8](#_Toc40707177)

[2.1 Project Goals 8](#_Toc40707178)

[2.2 Project Motivation 8](#_Toc40707179)

[2.3 Overview 8](#_Toc40707180)

[3 The Internet of Things Overview 9](#_Toc40707181)

[4 Project Architecture 10](#_Toc40707182)

[5 Development Platform and Tools 11](#_Toc40707183)

[5.1 JavaScript 11](#_Toc40707184)

[5.2 Node.js 12](#_Toc40707185)

[5.3 MongoDB 12](#_Toc40707186)

[5.4 AngularJS 13](#_Toc40707187)

[5.5 Express.js 13](#_Toc40707188)

[5.6 Arduino IDE / NodeMCU ESP8266 13](#_Toc40707189)

[6 Sensors 15](#_Toc40707190)

[6.1 Fingerprint sensor – JM-101 15](#_Toc40707191)

[6.1.1 Working principle 15](#_Toc40707192)

[7 The build 17](#_Toc40707193)

[7.1 Arduino Setup 17](#_Toc40707194)

[7.1.1 Enabling WIFI 18](#_Toc40707195)

[7.1.2 Fingerprint processing 19](#_Toc40707196)

[7.2 The Node.js web application setup 22](#_Toc40707197)

[7.2.1 The Node.js web application 23](#_Toc40707198)

[7.2.2 Programming the Web application 25](#_Toc40707199)

[8 Problem Solving 34](#_Toc40707200)

[9 Conclusion 35](#_Toc40707201)

[10 References 36](#_Toc40707202)

# Summary

The goal of my project is to design and build a software system to monitor staff in any small-scale business.

The idea behind this project comes from my many years working in the hospitality sector and from my internship I completed with SAP. During my time as a manager in the hospitality sector I was tasked with creating rosters and monitoring staff workflows. This was all paper based, and without the use of any software system. Moving forward to my internship with SAP I was able to see the benefits of these system and tasked myself to create a system that could be implemented in any small scale business as these are the business that would be still opting for the paper base route.

The Employee Management System (EMS) I have decided to create will consist of a web-based CRUD application. Administrators on the system will have the ability to create update and delete employees. These employees will be created using a fingerprint sensor that is connected to an ESP32, using HTTP post requests the data will be sent to the CRUD application that will be designed using the MEAN (MongoDB, Express, Angular and Node.js) stack.

Employees that are created on the system will only interact with the fingerprint sensor, once registered on the system the fingerprint sensor will be used to timestamp employee’s activity at work including arrival, breaks and departures from their job.

# Introduction

## 2.1 Project Goals

* To create a system to view and monitor employee’s and their workflows
* To include a unique identifier to distinguish employee’s
* Create an administrative web application to view these workflows and employees
* To include some form of IoT ideas implemented using a microcontroller
* To combine all aspect of the project to interact as designed/expected

## Project Motivation

The motivation behind this project stems from my experience working in the hospitality sector and from my internship at SAP. While many large-scale organisations have such applications to do this, for smaller organisations these systems are becoming more in demand. The use of technology has the potential to transform these business and aid in the overall success of said companies. From speaking to the owner of the hotel I have worked in “The Ocean Sands Hotel” he expressed his desire to have an application such as the one I have created.

## Overview

This report aims to give the reader a clear understanding of my designed system and the steps taken to create such a system. I will be outlining the procedure taken to develop the individual parts using the aid of diagrams where needed and briefly describing technologies used to assist me in complete this project.

# The Internet of Things Overview

The Internet of Things (IoT) refers to a network comprised of physical objects capable of gathering and sharing electronic information. The Internet of Things includes a wide variety of “smart” devices, from industrial machines that transmit data about the production process to sensors that track information about the human body.

How the Internet of Things Works

These devices use Internet protocol (IP), the same protocol that identifies computers over the world wide web and allows them to communicate with one another. The goal behind the Internet of things is to have devices that self-report in real-time, improving efficiency and bringing important information to the surface more quickly than a system depending on human intervention.

[[1]](#sourceA) Source: <https://www.investopedia.com/terms/i/internet-things.asp>

# Project Architecture

For my project I will be marrying hardware and software to be integrated to work together to make a fully functional project. I am using a Dolla tech jm-101 digital optical sensor connected to a NodeMCU ESP8266 board. This will communicate to the Node.js server via HTTP requests.

Beyond this the project is software based. This report will be coving the software aspect in the coming chapters.

A picture containing clock

Description automatically generated

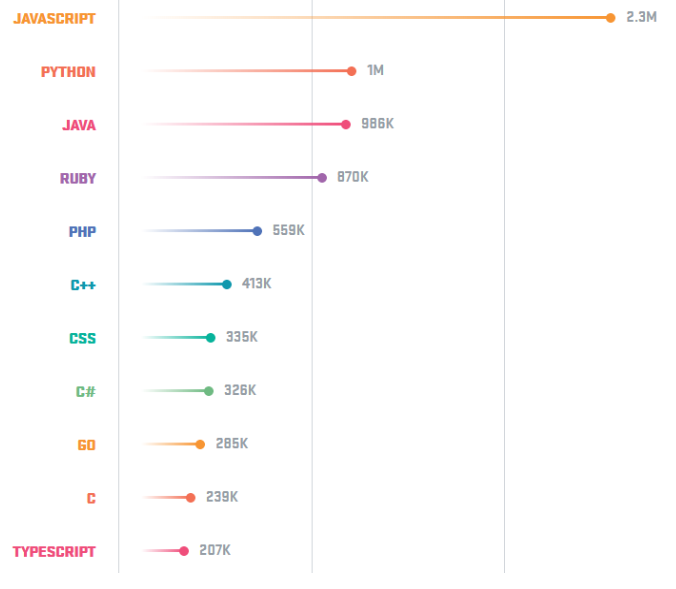
Figure ‑ Architecture Diagram

# Development Platform and Tools

## 5.1 JavaScript

JavaScript has often been described as “the language of the browser” created in 1995, a common misconception with JavaScript is it is related to java but for you I will clear up this idea. Put simply the only thing that these languages have in common is that sun’s JAVA was trendy at the time so the Netsacape was in a battle with Microsoft for popularity and Microsoft even released its own version of client-side programming called Jscript. in November of 1996 and JavaScript was standardized as ECMA-262 in June 1997. However, the name JavaScript is still used to this day.

JavaScript did not have an easy lifetime it was conceived by some to be lacking in performance and only used by amateur developers. However, the evolution of the language thanks to extensive time and energy the language has thrived over time. According to [[2]](#sourceB) [business insider](https://www.businessinsider.com/most-popular-programming-languages-github-2019-11?r=US&IR=T#1-javascript-10) JavaScript “remains the most popular programming language”



This stems from the transformations the language has taken in recent years, most notable being the introduction of node.js.

## Node.js

Node.js was created by Ryan Dahl in 2009 and is a framework to develop scalable network applications. Node.js is built on chromes V8 engine and allows the JavaScript to be used outside the browser. Node provided a full system JavaScript API that was never really achieved before due to the sandboxed environment that browsers provide. With the advent of Node, JavaScript developers could access the file system, open network sockets, and spawn child processes.

The characteristics of node as a language are

* Easy to use and implement
* Packs a lot of functionality into a small amount of code
* Large open source community
* Utilizes the NPM package manager
* Asynchronous and Event driven
* Callback functions

Thanks to its popularity and global involvement Node.js has a vast library of modules for different functionality. This aids the process of building applications both swiftly and to a standard that is of high quality.

Node also has other factors that assist in its popularity such as MongoDB a document-based database technology, Angular.js a front-end web framework maintained by google and Express, Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. The combination of these technologies even holds its own term the ‘MEAN’ stack.

## MongoDB

Mongo is a NoSQL structured technology, NoSQL databases store data in a variety of formats (documents or key value pairs) and are less structured than their relational database counterparts.

Mongo was designed by 10gen in 2007 and finally brought to market in 2009. This open sourced technology is a document orientated style that stores data in Binary JSON (BSON) documents. Due to using JSON, Mongo becomes ideal to read and write object from JavaScript code from what is a considerably low amount of code compared to the SQL based counterparts.

## AngularJS

AngularJS (version 1) or Angular (Version 2+) was conceived in 2009 by Miško Hevery and Adam Abrons. Angular owes much of its popularity to being backed by Google. It adopts the model-view-controller (MVC) approach to web applications. Angular provides two-way data binding between views and models also tasks can be created in augmented HTML.

## Express.js

Originally concepted by TJ Holowaycuk and released under the MIT License Express is a light-weight web application framework to help organize your web application into an MVC architecture on the server side. You can use a variety of choices for your templating language (like EJS, Jade, and Dust.js).

You can then use a database like MongoDB with Mongoose (for modelling) to provide a backend for your Node.js application. Express.js basically helps you manage everything, from routes, to handling requests and views.

## Arduino IDE / NodeMCU ESP8266

The Arduino IDE is a cross platform application that is written in C and C++. It us aimed to write code for Arduino compatible boards, however it can be implemented to work with other vendor boards as will be discussed.

The source code for the IDE is released under the GNU public license. The IDE supports the language C and C++ and consist of prewritten libraries to do several tasks with any compatible microcontroller. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

In this project the Arduino IDE is used to program the NodeMCU board. NodeMCU is an open-source firmware and development kit that helps to prototype or build IoT products. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The firmware uses the Lua scripting language. It is based on the eLua project and built on the Espressif Non-OS SDK for ESP8266

# Sensors

For this project the only sensor I will be using is a fingerprint sensor this sensor.

## Fingerprint sensor – JM-101

* The JM-101 fingerprint module is an integrated fingerprint processing module that integrates the optical path and fingerprint processing part.
* It has the characteristics of small size, low power consumption and simple interface
* It has high reliability, fast recognition speed, good wet and dry finger adaptability, and fingerprint search. high speed.
* JM-101 module communication interface is USB and UART communication interface.
* Dimensions (23.3\*20.3\*48.1) mm

### Working principle

**Fingerprint characteristics:**

The feature extracted by the fingerprint algorithm from the acquired fingerprint image represents the information of the fingerprint. Fingerprint storage, comparison and search are all done by manipulating fingerprint features.

**Fingerprint processing:**

Fingerprint processing consists of two processes: fingerprint login process and fingerprint matching process [where fingerprint matching is divided into fingerprint comparison (1:1) and fingerprint search (1: N)]. When the fingerprint is registered, each fingerprint is entered twice, and the recorded image is processed twice, and the composite template is stored in the module. When the fingerprint is matched, the fingerprint image is entered and the fingerprint image is verified and processed, and then matched with the fingerprint template in the module. If the template is matched with a template specified in the module, it is called a fingerprint comparison method, that is, 1:1. Mode; if matching with multiple templates, called fingerprint search mode, i.e. 1:N mode), the module gives the matching result (pass or fail).

**Technical parameters**

Supply voltage: DC 3.3V

Supply current: Operating current: <60mA

Peak current: <60mA

Fingerprint image entry time:

Window area: 15.3╳ 18.2 mm

Resolution: 500dpi

Backlight colour: blue

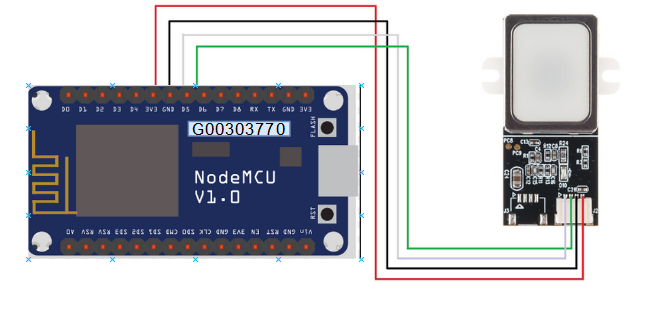


Figure ‑ Fingerprint Sensor

# The build

This section will take the reader through the whole build process and provide the steps needed in the development of my project.

## Arduino Setup

To set up the environment for the Arduino IDE to allow programming for the NodeMCU board there is a few configuration steps that need to be followed:

* Step 1

The Arduino IDE must be informed of the board that is being used. This is done by adding the board to the boards manager In “File -> Preferences” paste this link: <http://arduino.esp8266.com/stable/package_esp8266com_index.json>

* Step 2

Select the board, in my case I have chosen to select “NodeMCU 1.0 (ESP-12E Module)” board

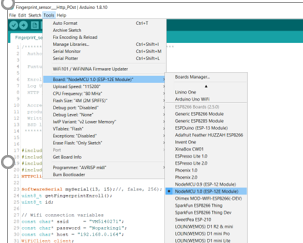


Figure 7‑ Board select

* Step 3

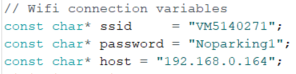
Following the instillation of the correct board we then need to download the libraries.

These include the [[3]](#sourceC) [Adafruit fingerprint library](https://www.arduinolibraries.info/libraries/adafruit-fingerprint-sensor-library) and the [ESP HTTP Client library](https://github.com/espressif/arduino-esp32/tree/master/libraries/HTTPClient).

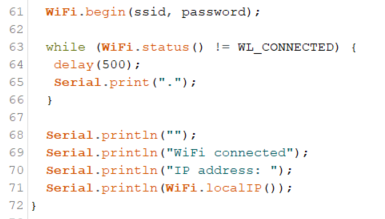
Once these steps are completed you now have a development platform configured to fit the requirements of this project. Next we move on to the programming.

### Enabling WIFI

The NodeMCU allows for connection to a WIFI network. To do this we must explicitly state the network and provide the wep key.



In the setup() function we connect to the internet by calling the wifi.being method. We can see if this connection is established by calling the wifi.status() method and check for success



Finally, when the conditions are met (registered employee sign in) we pass to a function called send employee. Here we send the data via http post request



As seen pictured above is the send employee function. Here we can see it is passed a variable called current ID. This ID is then attached to the URL that will be posting to the API. The server is configured to listen on this URL and will log the time of sign in.

### Fingerprint processing

As stated, we will be using Adafruit’s fingerprint library. The work in this section is adapted from the fingerprint enrol example given by Adafruit. “This library will let you use an Adafruit Fingerprint sensor on any UART to get, store, retrieve and query fingerprints!”. Included in the library are predefined functions that allow for easy integration with any optical fingerprint sensor.

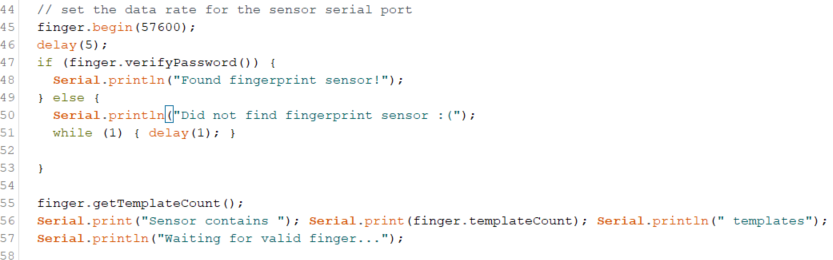
Throughout section 7.1.2 where used any Adafruit’s functions will be explained.

The sensor is connected via serial to the NodeMCU



We then need to create an instance of Adafruit\_Fingerprint called finger.

Following this the next step is to setup the fingerprint sensor



We start by setting the data rate for the sensor, then we move on to the if / else statement you can see the general idea for this is to check if there is a fingerprint sensor if yes we print a message in the serial window if there is no sensor detected we also inform the serial window and enter a while loop.

Then we call the sensor to read any stored templates.

Finger.verifyPassword this checks to see if the connection is correct, returns True/False.

Finger.templateCount this function requests the sensor to list of all template locations in use and stores them in self.templates. Returns the packet error code or OK success.

Moving to our regular operation of the sensor, simply we tell the sensor to search for prints constantly this is done by calling the getFingerprintIDez() function in the void loop.



This is the main function, p declared as an unsigned integer of the length 8 bits. Finger.getImage requests the sensor to take an image and store it memory, returns the packet error code or OK success. We then cycle through the checks to confirm if this template has been stored in the sensor. Upon success we inform the user via the serial monitor of success.

I have configured the system to work for general employees and this will send the data to the sendEmployee function. However, I need a way for an administer of the system to have the capabilities to add users to the system. This is done by assigning a template to the administrator in this case it is ID number 2. Then by calling the fingerprint enrol function.

Otherwise the ID gets passed to the sendEmployee function to send the data to the server.

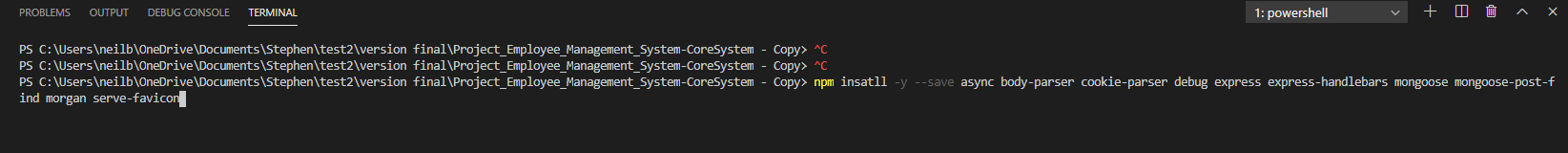
## The Node.js web application setup

First thing is to configure the system. Using the node package manager NPM we need to install all the modules that we require.

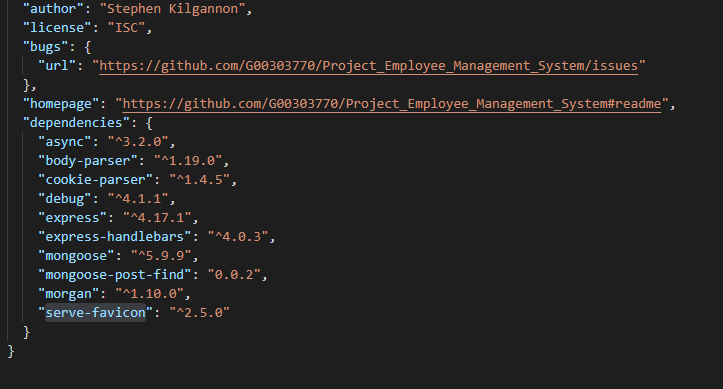
What is a module:

Consider modules to be the same as JavaScript libraries, a set of functions you want to include in the application. This project consists of core modules and user created modules.

To install these modules as stated we use the NPM



These modules then get written to the package.jason file



Package.json

All npm packages contain a file, usually in the project root, called package.json - this file holds various metadata relevant to the project. This file is used to give information to npm that allows it to identify the project as well as handle the project's dependencies. It can also contain other metadata such as a project description, the version of the project in a distribution, license information, even configuration data - all of which can be vital to both npm and to the end users of the package.

## The Node.js web application

Before moving to the code, we must first understand node.js and its programming model. Node.js implements event driven, non-blocking I/O model that makes it lightweight and efficient.

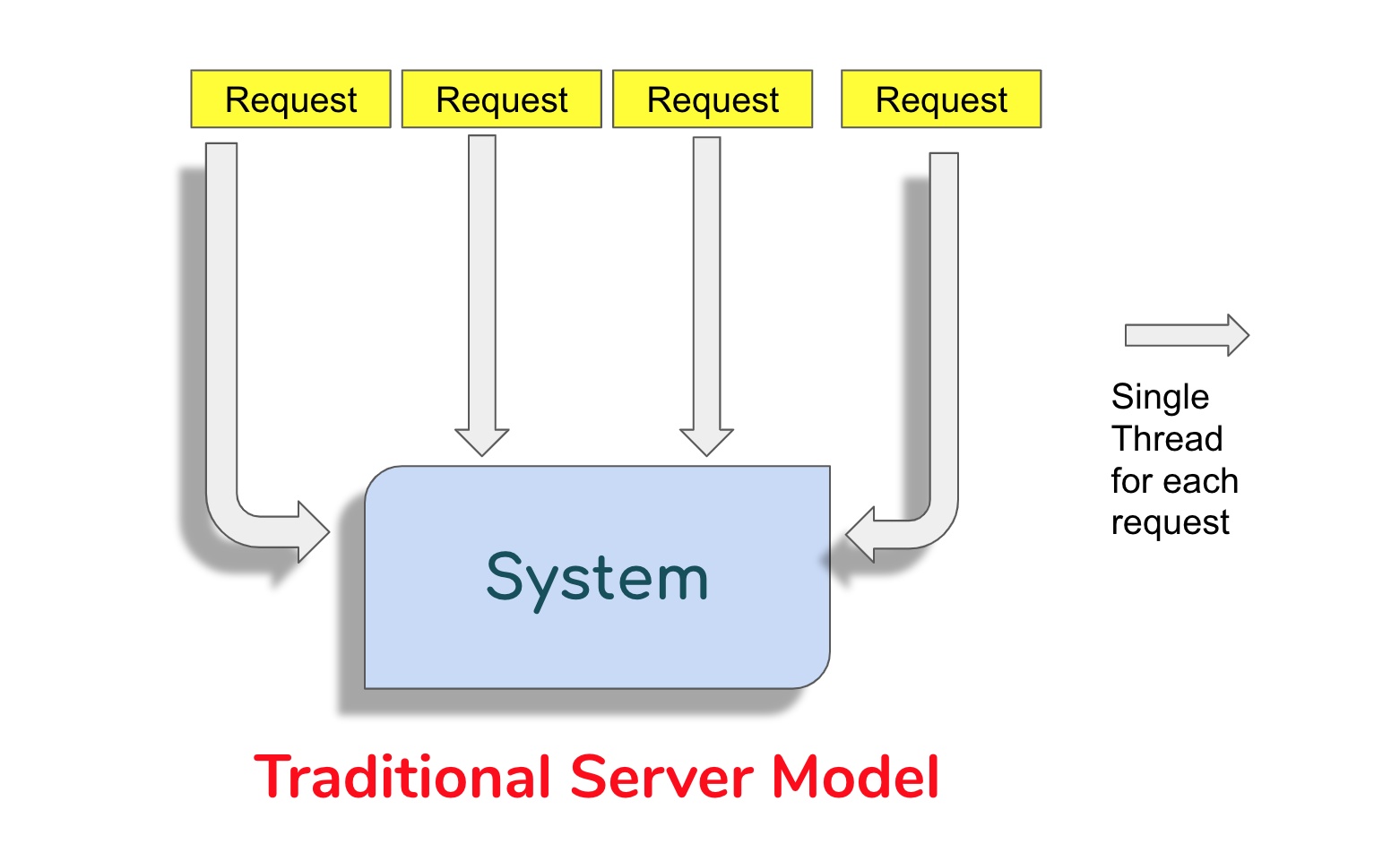


Figure 7-2 Traditional server model

In a traditional server requests are handled by assigning a thread. As multiple request come in the system will keep assigning threads this does not scale well. Imagine a system designed using this model. Imagine a scenario of a waiter in a restaurant take the waiter as the thread and the request is the customer this model would need a waiter for each table to do one task and to get more threads you would need more waiters hence more cost. This in hardware terms would require more computing power to handle the amount of threads.

So, would it not make more sense to have the waiter do multiple functions and when a job is complete to then handle the request? Yes, this is the way that it works in the real-world.

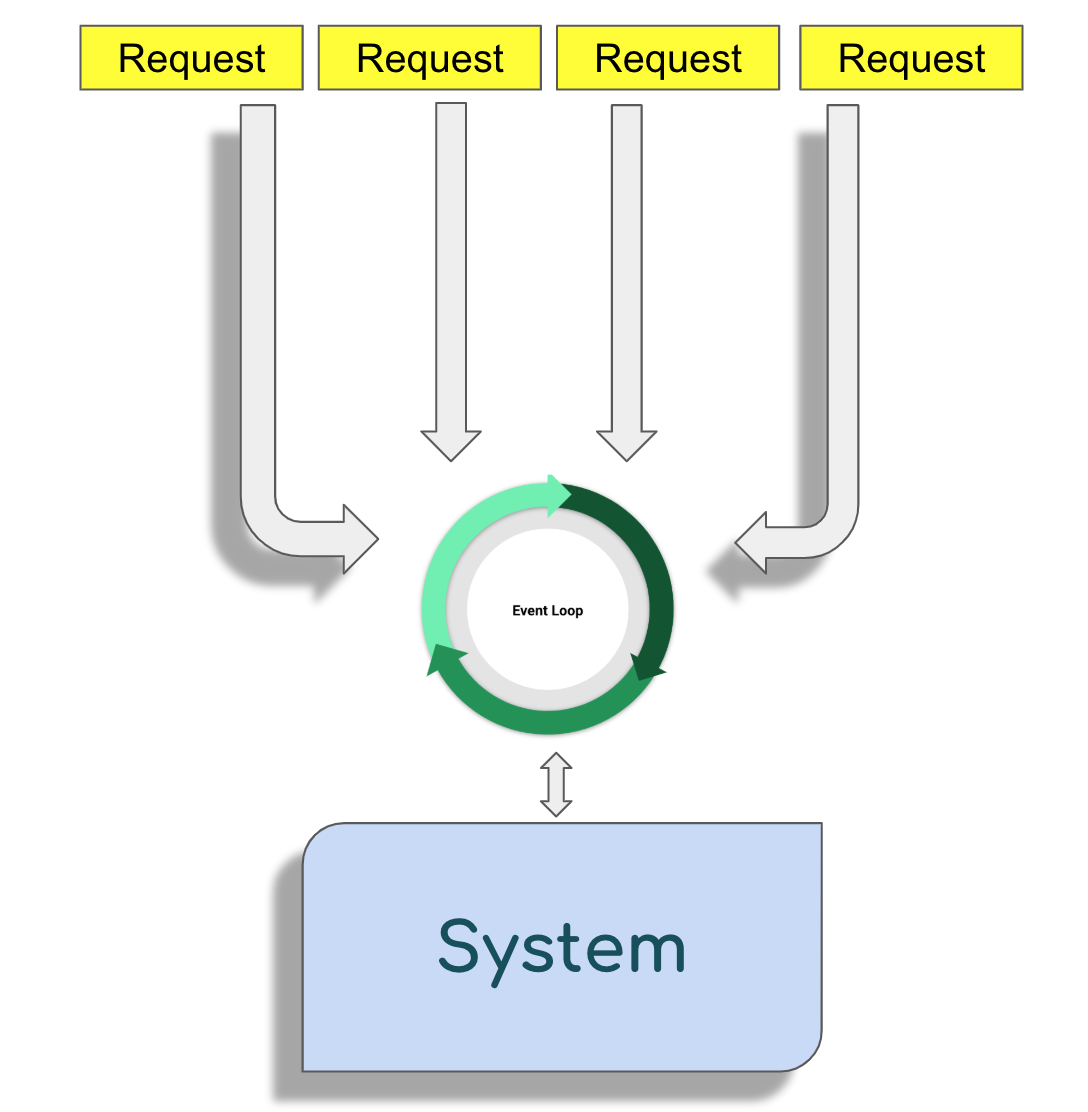


Figure 7-3 Node programing model

JavaScript and by extension node are single threaded. This means that JavaScript applications can only preform one task at a time, however by using the event loop it allows the application to give the illusion of being multi-threaded. How you may ask, well I had the very same question.

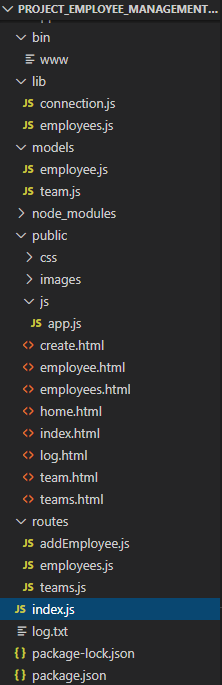
Features of Node:

1. Non-blocking
2. Asynchronous coding
3. Call-back functions
4. Event Emitters
5. Promises

Node is non-blocking which means it can take many requests at once. In Node programming model almost, everything is done asynchronously. Then we have call-back functions, a call-back function is a function that is invoked at the completion of an asynchronous operation with the result of the operation passed as function arguments. Event emitters then are used to create or publish an event such as a mouse click. Finally, promises to represent a value that is yet to be known. A promise can be thought of as a contract associated with an asynchronous function.

### Programming the Web application

Earlier we touched on modules, modules this is a key aspect in understanding this project I have written modules that will cover each technology in this project.

bin:

This is where the project is started from and requires index.js to be called, it does this by calling the require() function. This allows modules to be required in different aspects of the project.

Index.js:

Here we configure the middleware that will be used for express, express routes and MongoDB connection. Imagine index.js as the commander of the project.

Lib:

Here we configure the connection to the mongoDB database. Also, we have the code to allow the database to be searched for employees.

Models:

Here is listed the schemas used in the project.

Routes:

Here is the express of the application. Also, there is a add employee to add employees to the database.

Public:

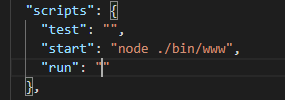
Everything for the front end or everything to do with angular.js is in this folder.

Package.json

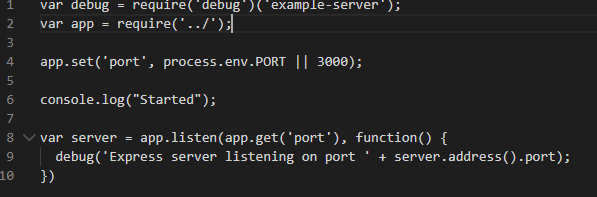
This list project dependencies and scripts to run the application

Now that the project structure is understood we can now delve further into the project.

The project is running from the bin folder this is done by defining a start script in the Package.json file.



Then moving to the bin folder, we go through the process to run the application



Debug is used to give a detailed description if an error is encountered. We then call the require function ../ states for the application to use any module in the root folder this being index.js. The we inform the server to be created and set to listen on port 3000.

**Index.js**

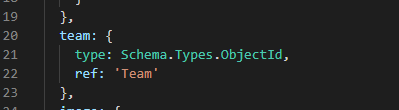
The file begins by importing express, path and some middleware. Next the connection to the database is established. The application routes are imported and initialize the app variable. Middleware (functions that execute during the lifecycle of the request to the express router)

**Lib**

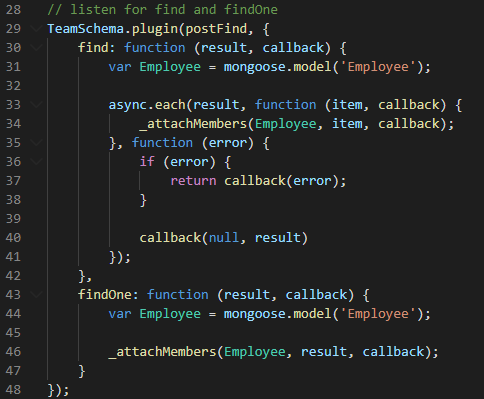
Connection.js establishes the connection to my cluster using the mongoose module. As you will see from looking at the code this is a very easy process compared to conventional SQL. Employee.js allows the employees to be sent when requested by the server.

**Models**

Employee.js holds the schema for creating employees and lists all the. Everything in this relatively straight forward. The team is simply a reference to an instance of the Team model.



Team.js schema is very easy to understand, the schema only has team name and an array of team members. Things get more complex around the added mongoose functionality



Plugin allows a mongoose schema to utilize a plugin to allow extended functionality. This is where mongoose-post-find is enabled to post find() and findOne() hooks. These hooks are functions that are run when find() and findOne(). These hooks can modify the results before returning to the calling function.

**Routes**

This project utilizes the express router. Here is where the routes are described.

Employee routes:

* GET /employees ― returns all employees, sorted by last name
* GET /employees/:employeeId ― returns information for a single employee, whose employee ID is passed through the employeeId parameter
* PUT /employees/:employeeId ― used to update an existing employee; the employee ID is passed through the employeeId parameter, while the employee data is passed in the request body

Team Routes:

* GET /teams ― displays all the teams stored in the database, sorted by team name
* GET /teams/:teamId — displays information regarding a single team; the team's ID is passed through the teamId parameter, and must match the associated MongoDB \_id property

Add Employee:

* POST /FingerPrint – looks for a url that gets passed from the ESP32. It then writes this information to the log file on the server
* PUT /addEmployees – Gets information to bea able to create new employees
* SAVE – saves the data to the MongoDB cluster.

**Public**

Here is housed all the angular code, this was the most difficult portion of the project

However, thanks to many resources online I was able to configure it to work with my project.

For styles I am using boots watch and for the fonts I used font awesome.

Single page applications (**SPA’s**) allow for pages to be dynamically loaded as needed. The characteristics that will needed to be known about SAP’s are:

* Ajax: Asynchronous JavaScript And XML allows web applications to be updated asynchronously by exchanging data with the express server. This allows the functionality to update certain parts of a webpage without reloading the entire page.



Figure 7-4 Ajax architecture [[4]](#sourceD)

* Templating: This allows for the server to update and convert JSON data into HTML that then can be injected into the webpage.
* Routing: just like on the server we had routing, when using angular you must use client-side routing.

**Model View Controller**

Angular implements model view controller to the front end.

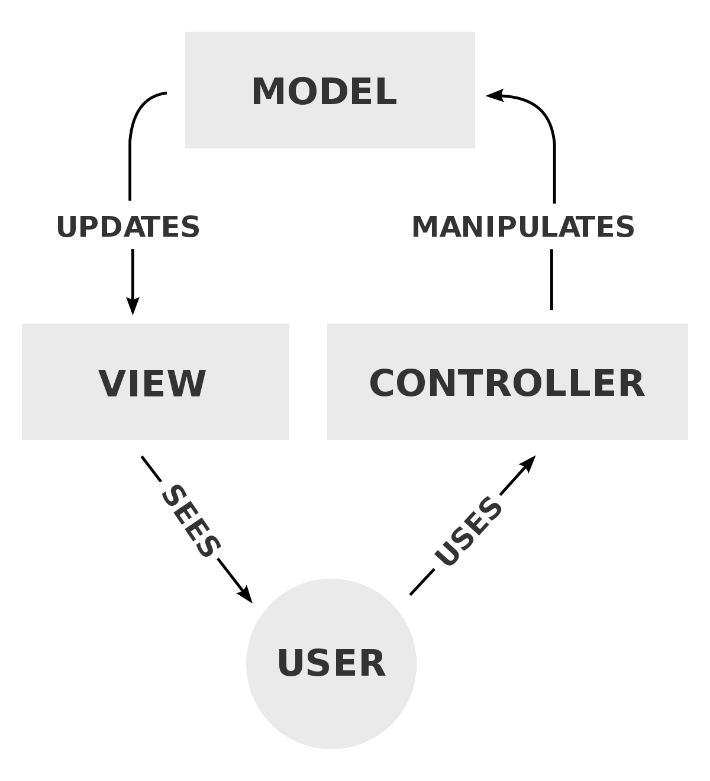


Figure 7-5 MVC architecture [[5]](#sourceE)

The model defines an application at the data layer, a controller is a JavaScript function that are configured in the app.js as controllers. The view is a visual representation of the model data these are defined in app.js as directives.

**Data Binding**

In programming, data binding is the automatic synchronization of data is between a program’s data layer and the associated view layer. Angular has two-way data binding a feature that is implemented to allow the employees to be updated.

Fundamentally, data binding consists of functions associated with a scope object (also called an Angular context). A scope object is a JavaScript object created with an Angular constructor that is used as the model for HTML mark-up. In a complete Angular application, the scope object is instantiated via a controller and made available to the view

**Directives**

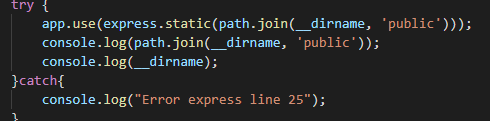
Directives are additional pieces of mark-up that Angular interprets to extend the default behaviour of HTML. Directives can be specified as custom HTML elements.

**Controllers**

Controllers are JavaScript functions that promote the use of code reusability. They are bound to the HTML. These functions are used to create a new Angular (or scope) context and are bound to sections of HTML mark-up. A controller provides a place to put application and view logic that is specific to a set of HTML tags. This is done by using ng.

Ng in angular applications this alerts the tag to be angular specific.

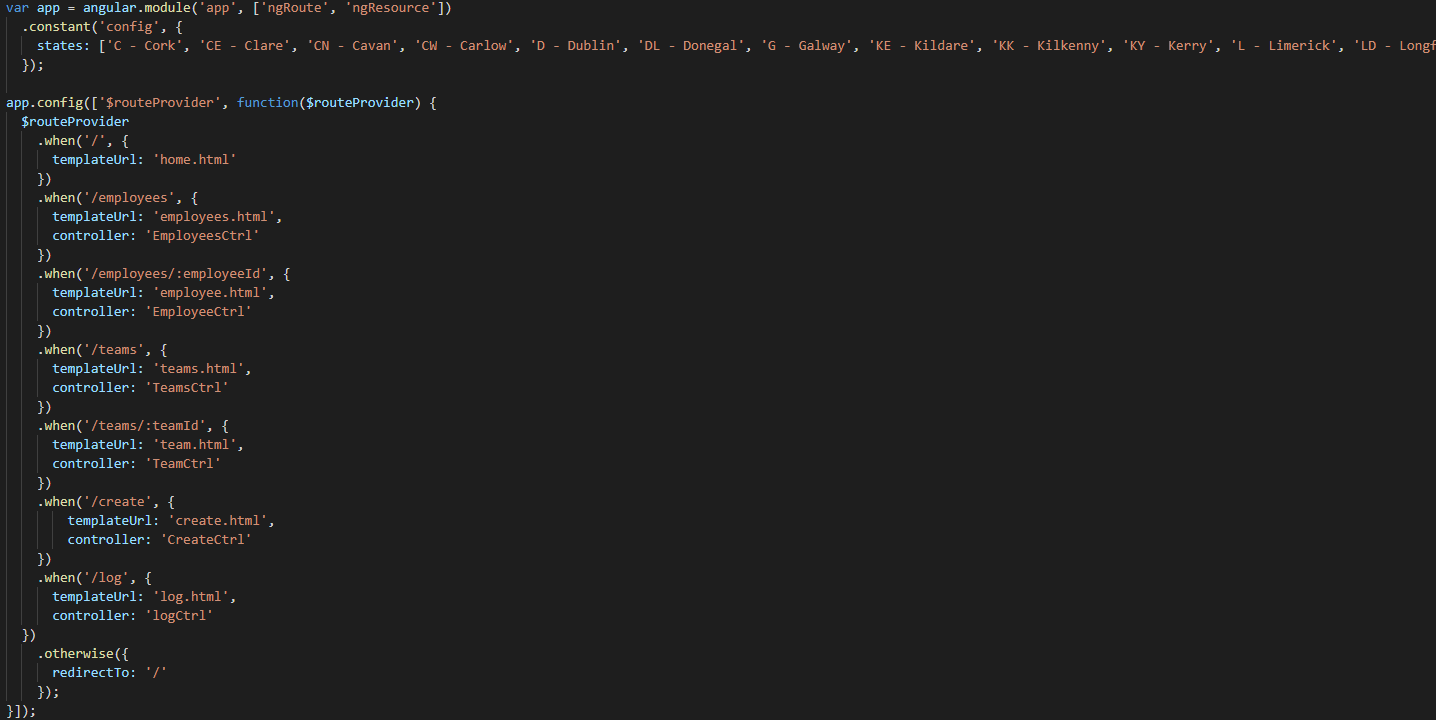
To configure the static middleware, we must inform the index.js file.



**App.js**

To begin we must declare the module named app, that is dependent on ngRoute and ngResource. A constant is then defined to list all the county’s in Ireland.

The config call is used to inject $routeprovider in order to create the client-side router. In this project there are 6 possible routes.



Then we declare the factory calls these are used to create the EmployeeService and TeamService. The factory resource depends on $resource.

Then there are the directives the image Fallback directive is used to apply the set an image to employees that is stored in the images folder. Then we move to the edit in line directive, this is how the user can edit the employee information without going through a direct route.

Finally, are the controllers these are JavaScript functions that handle when the application is provided a route. The EmployeeCtrl controller is used to work with the individual employee view page. This controller needs to perform several actions and relies on the q promises library. For example, this controller’s save() function is used to persist any changes to the MongoDB database.

# Problem Solving

Through out this project there was numerous issues I encountered. I found the angular code to be very difficult to begin however thanks to the books listed in the references I was able to incorporate this into the project.

I originally wanted the fingerprint sensor when enrolling a new ID to send this data the create portion of the application. This however ran into a dead end and due to time constraints, I had to revert to an alternative design.

Also due to angular being extremely volatile I on numerous occasions had difficulty loading the data to the application. However, after facing these issues implementing the single responsibility design approach while issues were difficult to fix, I knew exactly what portion of the project was failing.

The posting from the ESP32 caused me some issues to begin however this was due to following the original design. Once I reverted to the alternative design this stubbed these issues.

Finally, I would advise anyone who is creating web applications to use Firefox it has what can only be described as the best tool for web development. Learning to use the debugger completely changed how the outcome of this project.

# Conclusion

To conclude I am happy with the project I have created a web application that allows an administrator to edit, update and create employees. Also, I wanted to incorporate some IoT aspect in this project, this I achieved using the ESP to talk to the express server. The data from this is then read to a txt file located on the server.

I would have liked to stick with the original design for creating employees by using the #ID that is passed from the fingerprint sensor, I was able to implement a workaround on this but functionally it could be all incorporated and wold allow the flow to be much more as desired.

I logged all the employees sign in timestamps if I had more time to work on this, I would use this data to be displayed using charts.

Overall, I am delighted with the outcome, it implements most of the ideas that I wanted it to with a very few exceptions. I am looking forward to adding these functions to the project in my own time and have contacted the hotel I have worked in to see if this system would be of interest.

# References

[1] <https://www.investopedia.com/terms/i/internet-things.asp>

[2] <https://www.businessinsider.com/most-popular-programming-languages-github-2019-11?r=US&IR=T#1-javascript-10>

[3] <https://www.arduinolibraries.info/libraries/adafruit-fingerprint-sensor-library>

[4] <https://www.w3schools.com/xml/ajax_intro.asp>

[5] <https://en.wikipedia.org/wiki/Model–view–controller>

[6] Github <https://github.com/G00303770/Project_Employee_Management_System>