# Introduction

## Purpose

The following document is intended to give developers a better insight as to how the software will be developed from a technical perspective. Things like architecture, system design, component design, and all technical aspects of the software will be discussed. The documentation will strive to give a complete overview and ensure every developer is on the same page during the construction of the software product, “Help Me! Laurier”.

It should be noted that because of this, the documentation is **very technical.** If you are looking for a more broad requirements or introduction to the application, please consider reading the *Requirements* or *Analysis* sections first.

## Overview

To get a general overview of the document, see the table of contents provided. For a more in-depth look at what each section will contain, check out the below descriptions:

### System Overview

The system overview will give a general description of the functionality of the application. This will include things like: client-end choices, back-end choices, technology stack descriptions, database technology choices and a general overview of them. This section is designed so that a new developer can read it and figure out what their technical skill set will need to look like to contribute to the project successfully.

While the section will not outline every library and piece of software used throughout the system, it will describe all the major components. For example, if a specific server-sided library is used for a small task such as time zone calculations, it may not be mentioned. However, large dependent things such as **AngularJS** will be mentioned, as they are critical for the understanding of the entire application.

A brief description of why certain things were chosen will also be discussed along with other technologies that were considered. This will allow other developers to gain insight on how previous developers thought and why they had made the choices they had.

### System Architecture

The system architecture will provide a more detailed look at all the different subsystems and components of the application. Here, you will find decompositions of how each subsystem will interact, function and provide services to the other systems in the application. This gives a high level technical overview of the different systems in the application. If you are interested in how everything works from an overview perspective, this is the place to check out first.

In this section, you will also find some rationale for the choices that were made. This allows a section to document alternatives and notes on them. If a particular architectural decision is not going well, it is important to be able to reflect.

### Data Design

In this section, we discuss the overview of all data that flows in and out of the application. You will find a listing of all the data the applications needs to manage, how it is managed, where it is stored, processed and organized. If you want to know the specifics on a specific piece of data or how it interacts with the application, check here.

### Component Design

In this section, objects will be looked at it in a more individual level. The information from the system architecture overview will be analyzed in a more granular and specific way. Check this section out for brief descriptions on programming UMLs, member function descriptions, and descriptive code analysis.

## Further Reading / references

Before reading this document, it may be useful to list a few technical terms and information on where to read more about them as they are referred to throughout the documentation.

**Apache Cordova:** Apache Cordova is a set of tools designed to allow developers to package their web apps into native, easy to use packages to be distributed on various phone and tablet devices easily. (Read more: <http://cordova.apache.org/>)

**AngularJS:** AngularJS is a client side framework that makes writing MVVW like applications easy with JavaScript. (Read more: <https://angularjs.org/>)

**Ionic Framework:** The Ionic Framework is a high performance framework based on *AngularJS* and *Cordova* to create like-native applications using the greatest web application. (Read more: <http://ionicframework.com/>)

**SASS:** SASS is an extension of the CSS standard which describes some added in functionality, such as constants, expressions, and fallbacks for newer specifications. (Read more: <http://sass-lang.com/>)

**Bower:** A client sided package manager for libraries and tools. It used throughout the application to manage dependencies and ensure all developers are up to date. (Read more: <http://bower.io/>)

**npm:** The Node Package Manager (npm) is a tool for importing libraries and module into a node.js project. It is similar to Bower but is mainly used to manage the back-end of the software, rather than the client side. (Read more: <https://www.npmjs.org/>)

**V8:** V8 is Google’s implementation of the ECMAScript JavaScript specification. It is a fast, optimized version of the specification designed with performance in mind. (Read more: <https://developers.google.com/v8/>)

**Node.js:** Node.js is a sever side implementation of JavaScript using the V8 JavaScript engine to interpret. (Read more: <http://nodejs.org/>)

**MongoDB:** MongoDB is a document-oriented database that allows storage of data via blobs and “documents”, unlike traditional relational database management systems. It is a popular choice among many *Node.js* developers. (Read more: <http://www.mongodb.org/>)

**nodemon:** A build monitoring tool for *Node.js* that assists in rapid builds. (Read more: <http://nodemon.io/>)

**Gulp:** Gulp is a streaming build tool used to simplify the process of developing JavaScript applications. Similar to *make* in **C** and other build tools, *Gulp* can manage compiling, minifying, packaging, compressing images, building native applications with ***Cordova***and more. (Read more: <http://gulpjs.com/>)