COURSE INFORMATION AND BLOG

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Saurabh Shriniwas Kalantri
Spring 2016
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the
Thesis of Saurabh Shrinivas Kalantri:

COURSE INFORMATION AND BLOG

Carl Eckberg, Chair
Department of Computer Science

Tao Xie
Department of Computer Science

J. Carmelo Interlando
Department of Mathematics and Statistics

Feb 9, 2016
Approval Date
Copyright © 2016
by
Saurabh Shriniwas Kalantri
All Rights Reserved
DEDICATION

I would like to dedicate the work to my grand mother, my family and friends. They have been very helpful and also supportive and encouraging me to do the work in developing this application. I would also like to dedicate this thesis to my professor Dr. Carl Eckberg. Last but not the least, I would like to thank my friend Priyanka Manwani for helping me and guiding through development of this application, by providing the feedback.
ABSTRACT OF THE THESIS

Course Information and Blog
by
Saurabh Shriniwas Kalantri
Master of Science in Computer Science
San Diego State University, 2016

This project concentrates on developing a web application for not only prospective students but also for the students who are already admitted to our school. This application will give the overview of the courses taken in computer science department for Spring, Summer and Fall. In addition to that, student can also see the instructor who gave the course for particular semester. Student’s have the ability to post their queries and they can also see the question’s posted by other student.

In addition, a student can also see the professor’s who are in the computer science department and courses given by that professor in the current year and also they can see the previous year courses by going to courses category.

Technology wise, this application uses HTML5, CSS3, LESS, AngularJs to make the user interface of the application. It also uses Grunt, which is a building tool and NodeJs, which is the server side language. Also MySQL is used to store the data related to the application. In addition to that, GIT is used to maintain the repository of the application. As the application is based on agile methodologies, I had a discussion with professor Eckberg about developing the application in components, in which GIT helped a lot.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ABSTRACT</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>x</td>
</tr>
<tr>
<td><strong>CHAPTER</strong></td>
<td></td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td>1.1 Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Motivation</td>
<td>2</td>
</tr>
<tr>
<td>2 TECHNOLOGY</td>
<td></td>
</tr>
<tr>
<td>2.1 HTML</td>
<td>3</td>
</tr>
<tr>
<td>2.2 CSS3</td>
<td>4</td>
</tr>
<tr>
<td>2.2.1 CSS Box Model</td>
<td>4</td>
</tr>
<tr>
<td>2.2.2 Different CSS Modules</td>
<td>5</td>
</tr>
<tr>
<td>2.3 Bootstrap</td>
<td>6</td>
</tr>
<tr>
<td>2.4 Font Awesome</td>
<td>6</td>
</tr>
<tr>
<td>2.5 LESS</td>
<td>6</td>
</tr>
<tr>
<td>2.5.1 Variables</td>
<td>6</td>
</tr>
<tr>
<td>2.5.2 Mixins</td>
<td>7</td>
</tr>
<tr>
<td>2.5.3 Function</td>
<td>7</td>
</tr>
<tr>
<td>2.5.4 Comments</td>
<td>8</td>
</tr>
<tr>
<td>2.5.5 Importing</td>
<td>8</td>
</tr>
<tr>
<td>2.6 AngularJs</td>
<td>8</td>
</tr>
<tr>
<td>2.6.1 Model</td>
<td>9</td>
</tr>
<tr>
<td>2.6.2 View</td>
<td>9</td>
</tr>
<tr>
<td>2.6.3 Controller</td>
<td>10</td>
</tr>
<tr>
<td>2.7 GruntJs</td>
<td>10</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>HTML page structure [1]</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td>Box model [2]</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>Model view controller architecture [3]</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>Node js internal structure [4]</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>MySQL architecture [5]</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>Branch list and rule to create the branch</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Commit message Example</td>
<td>15</td>
</tr>
<tr>
<td>4.1</td>
<td>Course main page</td>
<td>17</td>
</tr>
<tr>
<td>4.2</td>
<td>Selection of semester</td>
<td>18</td>
</tr>
<tr>
<td>4.3</td>
<td>Detail about course selected</td>
<td>18</td>
</tr>
<tr>
<td>4.4</td>
<td>Course page template</td>
<td>19</td>
</tr>
<tr>
<td>4.5</td>
<td>Course page controller</td>
<td>20</td>
</tr>
<tr>
<td>4.6</td>
<td>Course page pop up template</td>
<td>21</td>
</tr>
<tr>
<td>4.7</td>
<td>Course popup page controller</td>
<td>22</td>
</tr>
<tr>
<td>4.8</td>
<td>Server-side code for course page</td>
<td>22</td>
</tr>
<tr>
<td>4.9</td>
<td>Professor main page</td>
<td>23</td>
</tr>
<tr>
<td>4.10</td>
<td>Professor pop up page</td>
<td>24</td>
</tr>
<tr>
<td>4.11</td>
<td>Professor page template</td>
<td>24</td>
</tr>
<tr>
<td>4.12</td>
<td>Professor page controller</td>
<td>25</td>
</tr>
<tr>
<td>4.13</td>
<td>Professor pop up page template</td>
<td>26</td>
</tr>
<tr>
<td>4.14</td>
<td>Server side for the professor page</td>
<td>27</td>
</tr>
<tr>
<td>4.15</td>
<td>Blog main page with question</td>
<td>28</td>
</tr>
<tr>
<td>4.16</td>
<td>Comment with question and comment</td>
<td>28</td>
</tr>
<tr>
<td>4.17</td>
<td>Blog pop up page template</td>
<td>29</td>
</tr>
<tr>
<td>4.18</td>
<td>Blog page controller</td>
<td>30</td>
</tr>
</tbody>
</table>
Figure 4.19. Comment page template.................................................................31
Figure 4.20. Blog page server side.......................................................................31
ACKNOWLEDGEMENTS

I would like to thank Dr. Carl Eckberg for his constant support and encouragement and guidance over the years without which this thesis would be incomplete. His patience and encouragement over the years has helped me immensely.

Additionally, my sincere thanks to the rest of my thesis committee: Tao Xie and J. Carmelo Interlando for not only serving as committee members but also for their encouragement and insightful comments.

I thank my fellow classmates for the discussions on the design of the application and all the fun we had with it during development of the application. Their feedback was most valuable.
CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Learning methods have evolved rapidly from book reading to learning through means of the internet, moreover we communicating with people via phone, email and social networking websites like facebook.com and twitter.com. With the boom in technology and increase in the use of internet learning, the internet medium has evolved to be one of the most important and easier methods in learning. This application combines online learning with online discussions about the application topics.

As this application is very helpful for the prospective students or the student who is already admitted to school as part of the computer science department. The main goal of this web application is to provide basic information about the courses taken in computer science at San Diego State University. Taking advantage of the boom in technology, this application will provide information to students with the help of the Internet.

International students who wants to be part of our school and want to get their Master’s Degree in Computer Science from San Diego State University have lots of question regarding courses given for the semester they are joining. Also as they contact professor through Email, this application will give a chance to instead post a doubt on the blog we provide in this application. With the help of this blog, not only can professor comment, but also seniors can provide their thoughts on the doubt or question asked by an individual. This can reduce the professor email volume.

Historically US has very good platforms for international students to expose their skills and contribute their talent to the world, so that student can emerge to produce many other platforms which will help other individuals in the world.

In addition, after a student has posted a doubt in the blog of the application, and if another likes the question or they don’t like the question, that student has the option to show
likes that particular question or dis likes the question by clicking one button in our application, so that if any other student who wants to see the question, they can discern the quality of question by seeing the likes on that question.

The report is divided into 5 chapters –

- Chapter 1: Introduction to the application
- Chapter 2: Technologies used in this application
- Chapter 3: Naming convention used in this application
- Chapter 4: Features in the application
- Chapter 5: Summary

1.2 MOTIVATION

Education is very important. In Fall 2013, when I applied for Master’s in Computer Science at San Diego State University, I was worried about the courses available for the semester Fall 2013. What I did was, I emailed many professors and messaged many seniors through facebook.com about my concerns on the courses. So at that time I thought that, I don’t want to flood the professor’s email with my question and also don’t want to make senior’s worry about my concern on their social sites. So I though of creating a web application or a platform for student like me who want to be a part of San Diego State University. Some course information is available in the SDSU Graduate Bulletin and at http://www.cs.sdsu.edu/prospectivestudents. This application at going beyond those resources and answering question like I had at my matriculation.

Also I have the inclination to help people, and is also a reason to build this application. Also when I attended my first class in fall 2013, I had many doubt about the assignments and other issues, so the application has the ability to post the question about whatever doubt they have. Also I had a problem getting the office location of the professor or the contact number of the professor, so this application has the ability to provide information about the professors in computer science department at San Diego State University.

However, understanding the problem I faced and thinking little bit of widely on what problems students can face in the near future, it motivated me to develop this application and provide the information student’s are looking for, as a kind of one stop shopping place.
CHAPTER 2

TECHNOLOGY

2.1 HTML

HTML (Hyper Text Markup Language) is the markup language, which describes the behavior of the web application.

- Markup language contains markup tags
- The html document or code is described by HTML tags
- Each HTML tag or element describes different document content

HTML tags are keywords surrounded by angle brackets:

\[
\text{<tagname>Content goes here</tagname>}
\]

- HTML tags normally comes in pairs \(<p>\) and \(</p>\), in which \(<p>\) is the start tag and \(</p>\) is the closing tag
- The start tag is also called opening tag and end tag is also called as end tag

HTML is a file with the extension .html, and Web Browsers like (Chrome, firefox, IE and etc) read HTML documents and renders them the end user. The web browser does not display the HTML tags but it display’s the content you write in between HTML tags.

```html
<html>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

**Figure 2.1. HTML page structure [1].**
The declaration of the HTML can be done as <!DOCTYPE HTML>, this declaration tells the browser the version which the HTML file is using. In this case it is using HTML5.

The common declaration for the HTML version’s are –

1. **HTML5** - <!DOCTYPE html>

2. **HTML 4.01** - <!DOCTYPE HTML PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN” “http://www.w3.org/TR/html4/loose.dtd”>

3. **XHTML 1.0** - <!DOCTYPE HTML PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN” “http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”>

As internet usage is grown rapidly, there are many HTML versions and HTML5 is the latest versions and this web application uses this version of HTML. The following are the html version and years.

- HTML - 1991
- HTML 2.0 - 1995
- HTML 3.2 - 1997
- HTML 4.01 - 1999
- XHTML – 2000
- HTML5 – 2014

### 2.2 CSS3

CSS stands for Cascading Style Sheet and 3 indicates the version of the CSS. It is the stylesheet language that defines the presentation of an HTML document. It describes how elements must show on screen, paper or in other media. It saves lots of work and it can control the layout of multiple web pages. CSS3 is completely backward compatible with earlier versions of CSS, it has all the properties and values which earlier version of the CSS were having.

#### 2.2.1 CSS Box Model

All HTML elements are considered as a box. In CSS, the term box-model is the layout of the particular HTML element. It consists of margin, border, padding and finally the content of the page.
Explanation of different parts:

- **Content** - It is where the text and image appear in the page
- **Padding** - It clears the area around the content of the page and it is transparent
- **Border** - A border goes around the content of the page and padding around the content.
- **Margin** – It clears an area outside the border and it is transparent as well

![Box model diagram](image)

*Figure 2.2. Box model [2].*

### 2.2.2 Different CSS Modules

- Selectors
- Backgrounds and Borders
- Text Effects
- 2D/3D Transformations
In our application we have used external css which is embedded as the stylesheet into the HTML head tag.

2.3 Bootstrap

Bootstrap is the most popular HTML, CSS and JavaScript framework to build a website which is mobile responsive. Bootstrap is a framework, which is completely free to use and free to download. It is also called a Front-End Framework, that is an interface for the user. That is, whatever the user sees on the page, that markup is designed and developed by the Bootstrap team.

2.4 Font Awesome

Font Awesome is a font, which is completely based on CSS and LESS. In this web application any icon you see about anything is from Font Awesome.

2.5 LESS

LESS is the CSS pre-processor, in which a less file is compiled to a CSS file. It extends the CSS and adds many features like variable, mixins, function and many other features, which makes CSS very easy to use and maintain. The web browser does not know about the LESS, it only knows about CSS. So it is mandatory that you convert or compile your LESS files into CSS. There can be a scenario where you have 10 LESS files and you don’t have to compile each LESS file into an independent CSS file; you can compile all LESS files into one CSS file.

2.5.1 Variables

Like other languages like Java, JavaScript etc., LESS gives us the ability to store the variable throughout the application. CSS does not have this feature. For example, In LESS if you want to declare the color #fff (White color), you can declare the variable as @white-color: #fff, so now you can use @white-color throughout your LESS files and then LESS gets compiled into CSS.
2.5.2 Mixins

Mixin is nothing but the class you have with some property and values; lets say if you want to use the same property and values to some different class, you can just insert the name of the class and LESS will automatically insert the property and values into the other class.

For Example –
CLASS - 1
.bordered {
    border-radius: 5px;
    border-color: 1px solid red
}
CLASS - 2
.some-feature {
    .bordered;
}

In above example you can see that, .bordered class has some styles and you have another class .some-feature and that class wants to use the styles in .bordered class; you can just insert the class name into the .some-feature class name. LESS will automatically take the styles from the class-1 and insert them into the class-2.

2.5.3 Function

Like other languages, the term function in LESS behaves the same. It accepts the argument or arguments. It provides a variety of functions, which transform colors, manipulate strings and do mathematical operations on that.

For example, if you want to use the border to the HTML element, you can use a function. The following code snippet shows how to use the function –
.class1 {
    border: .createBorder(1px solid red);
}
so I will create a function name createBorder

.createBorder (@x, @y, @z) {
    border: @x @y @z
This will lessen the work to write the border property elsewhere in the application. In that case you just call this function with the following three arguments.

### 2.5.4 Comments

In every project or every application commenting is a most important part of the code. If one developer wrote some code, and the other developer looks at the code, he would not have any idea if the developer has not commented on that. If the first developer has some kind of comment, then the other developer will know what’s going on with that particular code.

Like other languages, LESS also provides the commenting option.

For Example –

```less
//Some comment goes here or
/*Some comment goes here */
```

These are the two ways to comment in LESS.

### 2.5.5 Importing

If you have divided your project style component in different LESS files, you can import all the different component LESS files into one LESS file and then you can compile that one LESS file into CSS.

You can use `@import` keyword in LESS to import the .less files, followed by the filename in single or double quotes

For example

```less
@import 'variable.less'
@import 'course-core.less'
```

### 2.6 AngularJS

AngularJs is popularly known as Angular or Angular.js. It is an open source web application framework built on top of the JavaScript. As it is open source, any individual world can make changes according to his requirement. Basically the whole code base is maintained by Google Inc and by a community of individual developers and corporations to
address many challenges faced during the development of Single Page Application. It is an application, which provides a framework for client side model-view-controller.

![Model view controller architecture](image)

**Figure 2.3. Model view controller architecture [3].**

AngularJS’s design goals include –

- To separate DOM manipulation from application logic
- To separate client side of an application from the server side
- To provide the structure for making an application from designing the UI, to writing the business logic, to testing.

Angular implements the MVC pattern to segregate presentation, data and logic components.

### 2.6.1 Model

It is basically the data shown to the user in the view and the data with which the user interacts

### 2.6.2 View

Whatever the end-user see on the web page is called the view. It is the Document Object Model that the user sees on the web page and the content present in that object.
2.6.3 Controller

The controller handles the business logic behind what the user sees on the view.

In Angular, every project has one module and each module has one root scope and each module can have multiple views and each view is associated with a controller and every view or controller has its own variable scopes.

2.7 GruntJs

Grunt is a command line tool that runs on NodeJs and automates your JavaScript project build process. After installation of the GruntJs, you need to create the config file Gruntfile.js and create your task for processing your javascript project code base. You can create a watch task, using grunt modules watch, which watches for the file content change and processes the respective task if a watch task appears.

Also, if you are looking for a way to get all of your front-end code, ready for production with one command, a grunt is good. Basically, there are many grunt modules created by the grunt team, which you can use in your application. You can use lint, to lint your js files; you can use watch, to watch your js files, css files or html files, and you can also minify your js files into one js file. You can open your index.html in your browser (default browser).

2.8 NodeJs

Node.js is an open source, cross platform runtime environment used for development of the server side web application. It is an application that is written in JavaScript and can be run within Node.js runtime on any operating system platforms. The work done in Node.js is hosted and supported by Node.js foundation.

It uses Google V8 JavaScript engine to execute the code and a large percentage of the basic modules are written in JavaScript. It contains a built-in library to allow applications to act as a stand-alone web server. It allows the creation of web servers and networking tools using JavaScript and has a collection of different modules that handles various core functionalities. Modules like express, body-parser and many more are Node modules or npm modules.
Package management (npm – Node Package Manager) is the pre installed package manager for the Node.js server platform. NPM is basically used to install many of the node modules and also manages the third party Node.js programs. It is also used to create the REST API, so that we can exchange data from front-end and database using the REST API build with the help of Node.js.

The above diagram shows the internal structure or architecture of the Node.js. It shows that how libraries are used in JavaScript and the binding is used with the help of C, so that Node.js can communicate with the v8 JavaScript Engine.

2.9 Git

Git is the most widely used and most compatible version control system for the development of any software application. With most other version control systems and unlike most client-server systems, every Git working directory is a full-fledged repository with complete history of the individual file and full version tracking compatibility. Git is free software distributed under the terms and condition of the General Public License.
It also manages your source code, where you can make it private or make it public depending upon what application you are working on. If you are working on the open source it is advised that you make your repository public, so that many other developers can have a look at your git repository and if they are interested, they can contribute their thoughts as well. If you are working for your client, it is recommended that you held your repository private.

In our application, I have made the repository private, as it was work related to the school.

Git is also used to create the branch from the main source code, so that every developer can implement a new feature and merge the feature into the main branch, which is called “master”.

One of the advantage of creating a branch is, many developers can develop new features and merge different features into the master once they are fully developed by the developer.

2.10 MySQL

MySql is the open-source relational database management system (RDBMS). It is the most widely used relational database management system. It is the popular choice of database for use in web applications and is a main component of the widely used LAMP open source web application software stack. LAMP is a acronym for “Linux, Apache, MySQL and PERL/PHP/PYTHON”.

User can use the command line tool or use the MySQL workbench to create the data table and create some database rule. In our application we use MySQL Workbench to develop this application.

The following diagram shows the architecture of the MySQL
Figure 2.5. MySQL architecture [5].
CHAPTER 3

NAMING CONVENTION

3.1 NAMING CONVENTION FOR GIT

During the development of this application, I have maintained a naming convention while creating a branch; there is a commit message convention that is used. I have the list of the branch that was created during the development of this application.

3.1.1 Creation of Branch

“(Project Name)/(Project Name)-(Number)-feature-or-issue-description”

In this application, the project name is Course and the number is the feature number or component number for that particular branch.

The following figure will show the number of the branch created during development.

```
sdkasantrmbp:Course saurabh.kalantri$ git checkout master
Switched to branch 'master'
Your branch is behind 'origin/master' by 10 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)
sdkasantrmbp:Course saurabh.kalantri$ git branch
Course/Course-1-add-the-structure-to-the-project
Course/Course-10-ui-work-for-the-home-page
Course/Course-11-ui-work-for-professor-page-and-blog-page
Course/Course-12-make-the-course-page-dynamic-and-make-the-database-design
Course/Course-12-make-the-node-js-middleware-for-front-end-and-create-routes
Course/Course-13-make-the-professor-page-cone-from-back-end
Course/Course-14-add-ui-for-blog
Course/Course-15-make-the-course-page-dynamic
Course/Course-16-make-the-modal-work-for-professor-and-course-page-after-clicking-learn-more
Course/Course-17-add-the-backend-code-for-blog-make-the-database-design-and-make-the-ui-dynamic
Course/Course-2-create-the-grunt-automated-task-configuration-for-course
Course/Course-3-create-the-main-angular-module
Course/Course-4-add-injector-and-grunt-notify-task
Course/Course-5-add-the-less-directory-to-the-project
Course/Course-6-add-the-less-grunt-task
Course/Course-7-add-the-login-page-style
Course/Course-8-add-the-signUp-page-style
Course/Course-9-add-the-module-and-route-to-the-application

* master
```

Figure 3.1. Branch list and rule to create the branch.

In the figure you can see that there are 17 branches and you can see the consistency with which a branch is created. So basically this was the rule I followed so that I can keep track of the state of the application.
3.1.2 Commit Message

(Project Name)-(Number): type-of-the-component(Module Name). Your message goes here.

In this application, the project name Course and number is again the number of the feature and type is like feat – if you create a new feature, fix – if you are fixing the issue, refactor – if you are refactoring or optimizing the code.

The following figure will give you a glimpse of the commit message for one of the above branch is created during development of this application.

![Commit Message Example](image)

3.2 Naming Convention for JavaScript

In this project, there are many javascript files, to maintain some kind of rules for declaring variables and naming the files.

3.2.1 Variable Declaration

In this application all the variables are declared as a camel case declaration.

For example - var someVar = 10;

The way someVar is declared is called as camel case declaration.
3.2.2 File Naming

In this application every file is named as camel case declaration.

For example – loginCtrl.js

The way loginCtrl.js is named is called as a camel case declaration.
CHAPTER 4

FEATURES

4.1 Course

This feature is all about courses taken in the Computer Science Department at San Diego State University. In the application, if you select the Course category, you will have to select the semester, and after the selection of the semester you will see the courses offered in that semester, you can click Learn More to get more information about the course. The information will include a description about the course, the instructor of that course and also the location in which the class will be held, the timing of the class and also the days of the class.

There are two CTA’s (Call To Action) in this feature, Select Semester from the drop down, Learn More about the course.

The following are the screenshots and the descriptions related to that screenshots. Also you will see the code snippet for each feature.

4.1.1 Main Course Page

The following is the UI for main course page.

![Course main page](image)

Figure 4.1. Course main page.
4.1.2 Selection of Semester

The following is the UI for selection of the semester.

![Selection of semester](image)

Figure 4.2. Selection of semester.

4.1.3 Learn More About Selected Course

The following is the UI for learning more about the selected course.

![Course detail](image)

Figure 4.3. Detail about course selected.
4.1.4 Code Snippet for Course Page

The following figure will show you the template; it is nothing but HTML file, which has all the content in it.

Figure 4.4. Course page template.
The following figure is the Controller of the Course page; it has the entire event, like click, select from the drop down.

```javascript
'use strict';
var courseApp = angular.module('courseInformation');
courseApp.controller('CourseCtrl', ['$scope', '$rootScope', '$http', '$location', '$subModel', function ($scope, $rootScope, $http, $location, 
    $scope, init = function () {
        console.log("Control comes to Course Controller");
        console.log($location.path());
        console.log($scope.semesterValue);
        $scope.listOfSemester = ['Spring-2015', 'Summer-2015', 'Fall-2015'];
        $scope.semesterValue = 'Spring-2015';
        if ($location.path() == '/courses') {
            getCoursesFromServer($scope.semesterValue);
        }
    },

    $scope.onSemesterChange = function () {
        console.log($scope.semesterValue);
        getCoursesFromServer($scope.semesterValue);
    },

    $scope.onModuleName = function (parentIndex, childIndex) {
        console.log(parentIndex);
        console.log(childIndex);
        var finalDataToShow = $scope.courses[parentIndex][childIndex];
        console.log(finalDataToShow);

        var modalInstance = $subModel.open({
            animation: $scope.animationsEnabled,
            templateUrl: 'main/popups/views/coursePopUp.html',
            controller: 'PopUpCtrl',
            windowClass: 'courseModel',
            resolve: {
                data: function () {
                    return finalDataToShow;
                }
            }
        });
        modalInstance.result.then(function () {
            //
        });

    var getCoursesFromServer = function (semesterValue) {
        $http({
            method: 'GET',
            url: 'http://localhost:8000/getCourses?semester=' + semesterValue,
            headers: {
                'content-type': 'application/json'
            }
        }).success(function (data, status, headers, config) {
            console.log("I have a success");
            $scope.courses = Chukkida, data;
        }).error(function (data, status, headers, config) {
            console.log("Error comes here in courses");
        });

    };

    var chunk = function (myData, size) {
        var newData = [];
        for (var i = 0, length = myData.length; i < length; i += size) {
            newData.push(myData.slice(i, i + size));
        }
        return newData;
    };
```

Figure 4.5. Course page controller.
The following figure will show the pop up template; it is the template where you can see the detailed information about the course selected by student.

Figure 4.6. Course page pop up template.
The following figure shows you the controller which controls the data in the course pop up page template; it is the pop up where you can see the detailed information about the course selected by the student.

Figure 4.7. Course popup page controller.

The following figure is the server side code written in NodeJs, which serves the data from the database to the client side. In short, it is the way of communicating between Client Side and the database.

Figure 4.8. Server-side code for course page.
4.2 Professor

This feature will give information about the professor, their email id, their contact phone, their office location and also their research interests. It is very useful, for student who are looking for professor information, as they can just click on the professor category and by look in the list for the professor they are looking for.

In this feature, there are two CTA’s (Call To Action), so click on the Learn More to get detailed information about the professor; After clicking the Learn More you will see a pop up having information about the professor, and in case if you want to see the courses offered by the professor, you can trigger the CTA, to see courses given by the Professor selected by the student.

4.2.1 Main Professor Page

The following is the UI for main professor page in the application.

![Professor main page](image)

Figure 4.9. Professor main page.

4.2.2 Learn More about Selected Professor

The following is the modal for learning more about the professor in the department.
4.2.3 Code Snippet for Professor Page

The following page shows the template for the Professor main page; it is nothing but the HTML file which has some content, and this is the content student can see at the professor main page.
The following figure is the Controller for the Professor Page template; it means that any data that came from the server comes to this controller and then the controller will automatically update the view or template for the professor page.

```
'use strict;

var courseApp = angular.module('courseInformation');

courseApp.controller('ProfessorCtrl', ['$scope', '$rootscope', '$location', '$http', '$subModel', function ($scope, $rootscope, $location, $http, $subModel) {
    $scope.profile = function () {
        console.log('Controller comes to professor controller');
        $http({
            method: 'GET',
            url: 'http://localhost:8080/getProfessor',
            data: {semester: '234'},
            headers: {
                'content-type': 'application/json'
            }
        }).success(function(data, status, headers, config) {
            console.log('Have a success');
            $scope.profiles = chunk(data, 2);
        }).error(function(data, status, headers, config) {
            console.log('Error comes here in professor');
        });
    }
    $scope.getProfessor = function () { }
    $scope.selectLearnMore = function (parentIndex, childIndex) {
        console.log(parentIndex);
        console.log(childIndex);
        var finalDataToShow = $scope.profiles[parentIndex] [childIndex];
        console.log(finalDataToShow);
        var modelInstance = $subModel.open({
            controller: 'ProfessorPageModel',
            templateUrl: 'main/views/professorPage.html',
            windowClass: 'professorModel',
            resolve: {
                model function() {
                    return finalDataToShow;
                }
            }
        });
        modelInstance.result.then(function () {
            }
        });
        var chunk = function (myData, size) {
            var newData = [];
            for (var i = 0, length = myData.length; i < length; i += size) {
                newData.push(myData.slice(i, i + size));
            }
            return newData;
        }
    }
});
```

**Figure 4.12. Professor page controller.**

The following figure is the template for the professor pop up page. It is nothing but the pop up, when you click the learn more button on the main page of the professor. The pop up shows the detail information about the professor selected by the student and also the student can see the courses given by that professor by triggering a click to see the courses given by the instructor. The controller for the pop up page is the same as the pop up controller for the course pop up page.
Figure 4.13. Professor pop up page template.

Figure 4.14 is the server side code written in NodeJS. This code serves the data from the database to the client side, that is, whatever user see’s on the page is with the help of the server side code.

4.3 BLOG

This feature in the application is for the students who have many doubts and wants either the professor to answer the doubt or allow any other student who really want to comment on that doubt, so it has a feature to post your doubt on the application. Also a student can also see the questions posted by other student, so that they have an idea about what other student doubts are. In addition to that, a student or professor or any individual looking at the question can like or dislike the question. Moreover by doing that if student comes and browse through the entire question, the responses can be sorted based on how
many likes are there on the question. So student can read first the questions that have more likes and ignore those which has more dis likes.

### 4.3.1 Blog Main Page with Question

The following is the UI for blog main page question, where student can see the question posted by other students.
4.3.2 Blog Main Page with Question and Comments

The following UI show the comments on the question posted by the student.

Figure 4.15. Blog main page with question.

Figure 4.16. Comment with question and comment.
4.3.3 Code Snippet for Blog Page

The following figure is the template for the Blog Page, where students can post their questions and also students can see the questions posted by other students. They can also like the question or dislike the question as well.

Figure 4.17. Blog pop up page template.

Figure 4.18 shows the controller for the Blog Page template; it means any data coming from the server comes to this controller and then the controller serves the data to the template. This controller is the same for the Comment template of the application.

Figure 4.19 is the template for the comment. It is the template where you can see the entire comment on that question. It is triggered when the user clicks on see all comments. This template is included into the blog main page.

Figure 4.20 is the server side code written in NodeJs for the Blog Page. It is basically used to serve the data from the database to the client side, so that the user can see the information they are looking for.
```javascript
'use strict';

var courseApp = angular.module('courseInformation');

courseApp.controller('BlogCtrl', ['$scope', '$routeParams', '$http', function ($scope, $routeParams, $http) {
    $scope.list = function () {
        console.log('Get all the blog posts!');
        $scope.data = [];
    };

    $scope.onSubmitQuestion = function () {
        console.log($scope.data.question);
        $scope.submitted = true;
        if ($scope.questionForm.$valid) {
            if ($scope.data.question) {
                $scope.data.question = $scope.questionForm.$valid;
                $http(
                    method: 'POST',
                    url: 'http://localhost:8000/addQuestion',
                    data: $scope.data.question,
                    headers:
                        { content-type: 'application/json' }
                ).success(function (data, status, headers, config) {
                    console.log('I have a success!');
                    $scope.data.question = data.result;
                }).error(function (data, status, headers, config) {
                    console.log('I have an error!');
                });
            }
        };
    };

    $scope.showComments = function (index) {
        console.log(index);
        $scope.showComments = true;
        $scope,index.Show = index;
        var question = $scope.question[index];

        $http(
            method: 'GET',
            url: 'http://localhost:8000/getComments?questionId=' + question.id,
            headers:
                { content-type: 'application/json' }
        ).success(function (data, status, headers, config) {
            console.log('I have a success!');
            $scope.commentsList = data.result;
            $scope.showComments = false;
            $scope[index].ShowComment = false;
        }).error(function (data, status, headers, config) {
            console.log('I have an error!');
        });
    };

    $scope.updateComment = function (index) {
        console.log(index);
        var questionId = $scope.question[index].questionId;

        $http(
            method: 'POST',
            url: 'http://localhost:8000/updateComment',
            data: questionId,
            headers:
                { content-type: 'application/json' }
        ).success(function (data, status, headers, config) {
            console.log('I have a success!');
            $scope[index].like = data.result; // query to get the updated like
        }).error(function (data, status, headers, config) {
            console.log('I have an error!');
        });
    };

    $scope.deleteComment = function (index) {
        console.log(index);
        var questionId = $scope.question[index].questionId;

        $http(
            method: 'POST',
            url: 'http://localhost:8000/deleteComment',
            data: questionId,
            headers:
                { content-type: 'application/json' }
        ).success(function (data, status, headers, config) {
            console.log('I have a success!');
            $scope[index].like = data.result; // query to get the updated like
        }).error(function (data, status, headers, config) {
            console.log('I have an error!');
        });
    };
}
```

Figure 4.18. Blog page controller.
Figure 4.19. Comment page template.

Figure 4.20. Blog page server side.
CHAPTER 5

SUMMARY

5.1 CONCLUSION

During development of this application, I got an excellent opportunity to learn new technologies and get in depth knowledge about the technologies I already had an idea about.

Course Information And Blog, I think this is good platform for the student who really wants to do an M.S. from San Diego State University. This application provides the data for the student and on the basis of that data they can make a decision as to whether to go with Master at San Diego State University or some other organization. In addition, this is also a very good platform for student to discuss their question. Moreover, this application is developed in the very latest technology and will work on all modern browsers. Also this application is very lightweight and also has a good user experience.

5.2 OBSTACLES FACED

During the development of this application there were many challenges faced. One of the main problem was, I didn’t have much of the experience in Databases, so I had to talk to my friends and get an idea about how to design the database. Also, I did not have much experience about the server side Cross Origin Resource Sharing (CORS), so my front end was pointing to the wrong server and my server was pointing to a different host, So finally after two days of research I found that I need to allow the cross origin in the server side as javascript does not support cross origin.

5.3 FUTURE ENHANCEMENT

During development of every application, the developer plans on how can we enhance the application by developing new features.
The following is a list of improvements –

- To implement user based application
- To add chat option so that student can get the answers on the go
REFERENCES


