THE CE SHOP

Practical Interview Project

Position: Jr. Full Stack SoftwareEngineer

Table of Contents

[1. INTRODUCTION 3](#_Toc68418181)

[1.1. Analysis 3](#_Toc68418182)

[1.2. Customer/Student Role: 3](#_Toc68418183)

[1.3. Administrator Role: 3](#_Toc68418184)

[1.4. System Advantages 3](#_Toc68418185)

[1.5. Assumptions 4](#_Toc68418187)

[2. SOLUTIONS AND FEATURES 4](#_Toc68418188)

[3. SYSTEM ANALYSIS 5](#_Toc68418189)

[3.1. Functional Requirement 5](#_Toc68418190)

[3.2. Non-Functional Requirement 5](#_Toc68418191)

[4. TECHNICAL DESIGN 6](#_Toc68418192)

[4.1. Requirement Analysis/Technical Design 6](#_Toc68418193)

[4.1.1. Use Case and Activity Diagram: 6](#_Toc68418194)

[4.1.2. Class Diagram 8](#_Toc68418195)

[4.1.3. Sequence diagram: 9](#_Toc68418196)

[4.1.4. Data Dictionary: 9](#_Toc68418197)

[5. IMPLEMENTATION 12](#_Toc68418198)

[5.1. Tools/Platforms & Technologies 12](#_Toc68418199)

[5.2. User Interface Screenshots 12](#_Toc68418200)

[5.3. Code Implementation 14](#_Toc68418201)

[6. RECOMMENDATIONS 17](#_Toc68418202)

List of Figures

[Figure 1:Use Case Diagram for tracking student progress in a course 6](#_Toc68509106)

[Figure 2: Activity Diagram for tracking student progress in a course 6](file:////Users/home/OneDrive/Dollu/JobInterview/CE%20Shop%20Project%20Report%20-SA%20FEedback%202.docx#_Toc68509107)

[Figure 3:Class Diagram for tracking student progress 7](#_Toc68509108)

[Figure 4: Sequence Diagram for tracking student progress………………………………………………....................8](#Figure4)

Figure 5: Fetching the courses taken by student……………………………………………………………………...11

[Figure 6: Monitoring student progress throughout the course…………………………………………………… ….12](#Figure6)

List of Tables

[**Table 1: CE\_STUDENT**](#_Toc68509128) 9

[**Table 2: CE\_COURSE**](#_Toc68509129) 9

[**Table 3: CE\_MODULE** 10](#_Toc68509130)

[**Table 4: CE\_INSTRUCTOR** 10](#_Toc68509131)

[**Table 5: CE\_QUESTION** 11](#_Toc68509132)

# **INTRODUCTION**

The CE Shop offers online courses in Pre-Licensing, Post-Licensing, and Continuing Education (CE) for real estate agents in all 51 jurisdictions in the U.S and Pre-Licensing courses in 13 states. This report aims to provide solutions with respect to addition of a new feature in the Learning Management System developed by the CE Shop that will display the percentage of course completed by a particular student, for instance if 4 out of 10 questions are completed for a course, the system will display course completion progress as 40%. The proposed solution in this report enables the students to access the overall progress for a particular course taken by them, keeping a track of number of questions completed across all modules per course and displaying the results as student’s progress on the student profile page. Students can mark any question as completed or pending which will update the student progress details in the real-time.

The primary objective of this project is to analyze, design and develop a new feature for the learning management system for the CE Shop for displaying the progress made by student in a particular course. It aims at making the user interface easier for the students to track their course completion progress in real-time. The web application will include features such as displaying course details, accessing modules for course, updating question completion status, displaying the percentage of completed course, and generate transcript/certificate on course completion.

# **Analysis**

This project will add a feature for displaying the progress made by the student in each module of the current course being viewed. The student will be able access the modules based on course-id for each course and can update the completion status for each question for each module via a simple-click of a button. The details of courses purchased by a student, and list of modules for each course will be stored in a backend database maintained using Oracle database management system. This data will be further used to generate a certificate/ course transcript on the completion of course. This system has two types of users:

# **Customer/Student Role:**

The customer/student can login/logout the system and view current courses, modules per course, and update module completion status. The customer can just view the information whereas he/she cannot make any changes in the database.

# **Administrator Role:**

The administrator manages the entire database and can add/delete the courses from the database, monitor the course-completion status for student, and send notifications related to course deadlines for the pending modules and generate transcript/license certificate for student on completion of course.

# **System Advantages**

* + 1. Enables students to track the progress of courses taken by them for each question of the course modules in real-time.
    2. Enables administrator to monitor the student progress in course, send notifications regarding deadlines, and generate course transcript/certificate.

# **Assumptions**

* + 1. The system consists of an administrator to manage system and registered students taking courses.
    2. The system consists of courses which students can take.
    3. The system has registered instructors to teach courses.
    4. The courses consist of modules, which in turn consists of lectures/assignments.

# **SOLUTIONS AND FEATURES**

To monitor the progress of course completed by the student in real time, this solution creates a comprehensive platform with features to view the percentage of module completed for each course and update the completion status for each lesson of a module. The system tracks the completed and uncompleted modules and displays them as progress report in form of percentage to the student. This solution provides following features:

* Enables students to access questions for each course module.
* Enable students to mark a particular question for a module to be completed/done on completion.
* Monitor the completion status of the modules and update the completed modules once the status of a module reaches 100%.
* Enable the students to download certificate of completion and generate course transcripts on the course completion.

# **Plan of Work**

* + - Abstract realization
    - Gather requirements (functional/non-functional) of the project
    - Design database schema and UML diagrams for system design
    - Software implementation of modules:
      * Designing front-end forms of modules using Angular 8 and Bootstrap.
      * Programming for back-end in spring boot application.
      * Designing databases for our system.
      * Understanding user interaction with website module-by-module.

# **SYSTEM ANALYSIS**

# **Functional Requirement**

In order to add the new feature that tracks student’s progress for a course, the following major functional requirements were discovered:

* + 1. **Sort Courses by their expiration date**
       - * Fetch the list of courses taken by the student based on expiration date.
         * Update the status of courses past the expiration date as archived, and display in the archived section.
    2. **Sort Modules by Courses**
       - * Fetch the active courses for the student form the list of courses fetched previously.
         * Display list of modules in the courses when a course is selected by students.
    3. **Track the progress of Course**
       - * Update the progress percent of questions for each module, when all the questions are completed.
         * Update the percent of course completed as the percentage of completed questions divided by the total questions across all modules.
         * Generate the student transcripts and completion certification on completion of course.

# **Non-Functional Requirement**

The non-functional requirements help keep track of the operational, performance, and security of the new system.

* + 1. **Operational Requirements**
       - * 24\*7 access to the online courses via a stable internet connection.
         * The system should be compatible with all types of OS-Windows, MacOS, Linux, Android and IOS.
         * System should be able to perform on all types of devices such as PC’s, laptops, PDA’s, etc.
    2. **Performance Requirements**
       - * Monitor the progress made by the student for a module within the course in real-time.
         * Update the overall progress made by student for a course, based on number of modules completed.
    3. **Security Requirements**
       - * Enable only authorized student users to access the courses.
         * Enable multi-factor authentication along with login to authenticate the user identification.
    4. System Requirements
       - * **Hardware Requirements:** Desktop/Laptop devices, mobile devices, PDA’s, and any hardware required to connect to internet, e.g. Modem, WAN – LAN, Ethernet Cross-Cable.
         * **Software Requirements:** Linux, Windows, MacOS, Android and IOS compatible devices

# **TECHNICAL DESIGN**

# **Requirement Analysis/Technical Design**

# **Use Case and Activity Diagram:**

The actors involved are the registered students and the system.

The student actor performs following activities: -

1. View Registered Courses: Get a list of active/archived courses for student.
2. Select Course: Select a course from list, and fetch is modules
3. Complete Questions: Complete questions for each module
4. Track Course Progress: Track percentage of completed questions from total questions.
5. Complete Course: Complete all questions across all of the course modules
6. Download Transcript/Certificate: Receive grades and download certificate.

The CE shop system performs following activities for tracking the course progress:

1. Update Course Status: Update the status of completed questions
2. Calculate Course Percentage: Find the percentage of questions completed divided by total questions across all modules
3. Display Course Progress: Update the course progress on Student Profile



Figure 1:Use Case Diagram for tracking student progress in a course

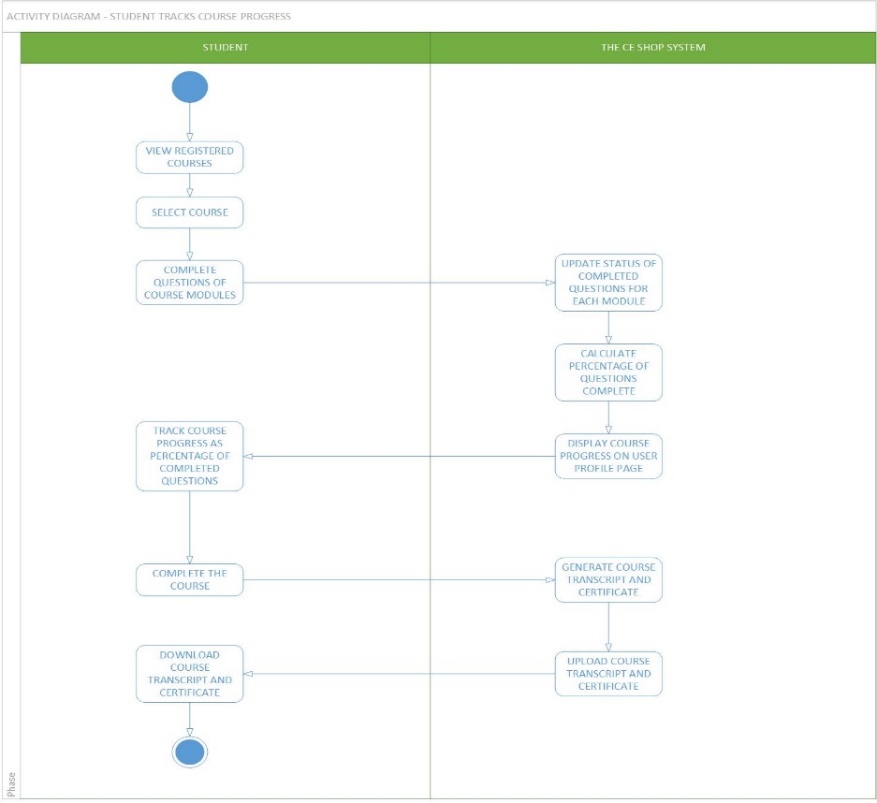


Figure 2: Activity Diagram for tracking student progress in a course

# **Class Diagram**

The following classes will be implemented:

1. Student: This class stores the information of students registered with The CE Shop.
2. Course: This class stores the information about courses offered by The CE shop.
3. Module: This class stores information regarding modules of a course.
4. Instructor: This class stores information about instructor teaching a course.
5. Question: This class stores information related to questions in a course module.

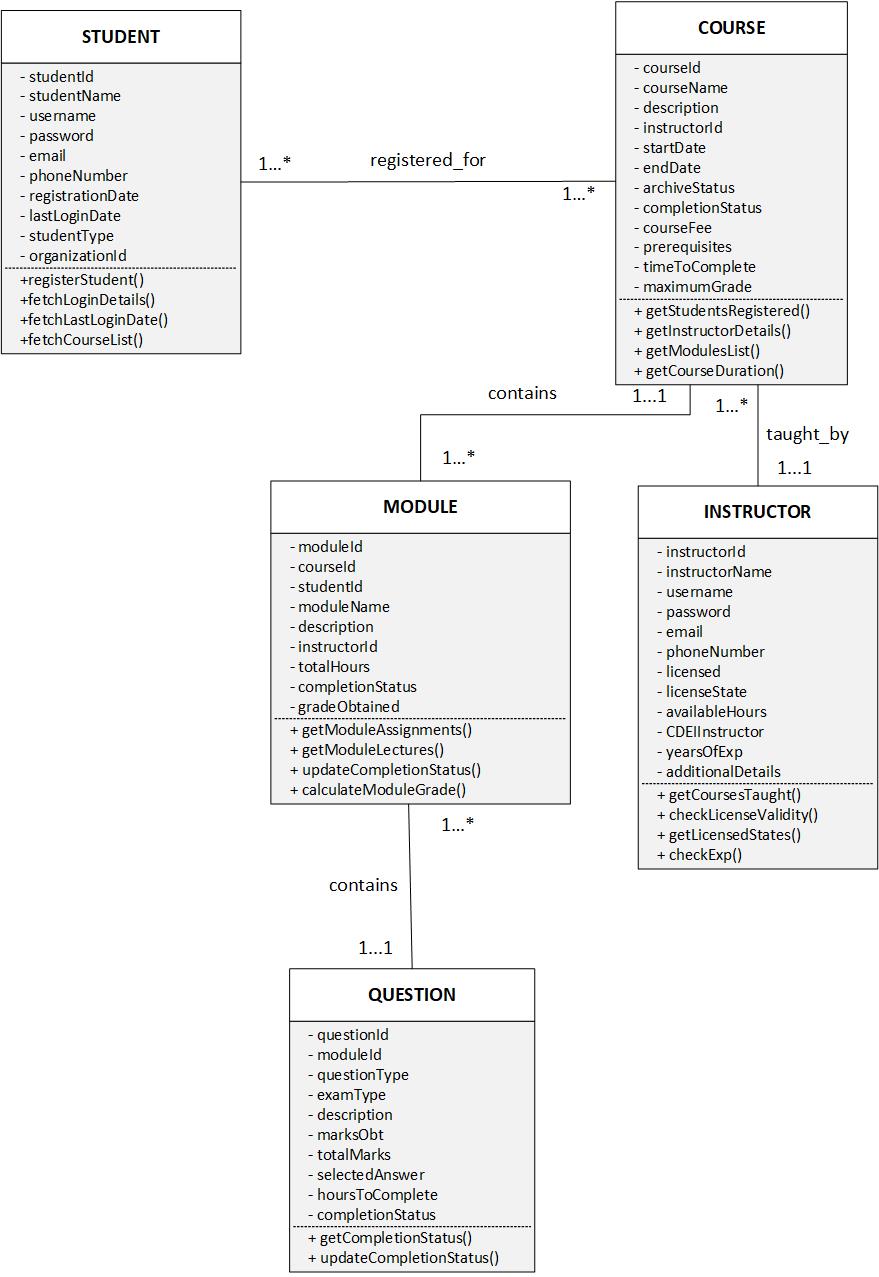


Figure 3:Class Diagram for tracking student progress

# **Sequence diagram:**

The feature will be implemented in the following sequence:

1. Student retrieves the list of registered courses and selects one course.
2. Student views the modules for the selected course.
3. Student completes questions for each module.
4. System updates the status of completed questions for course.
5. System calculates the percentage of course completed (completed questions/total questions).
6. System displays the percentage of course completed on student profile.
7. Student completes all questions for current course.
8. System updates the completion status of course and generates transcripts/certificate for student.

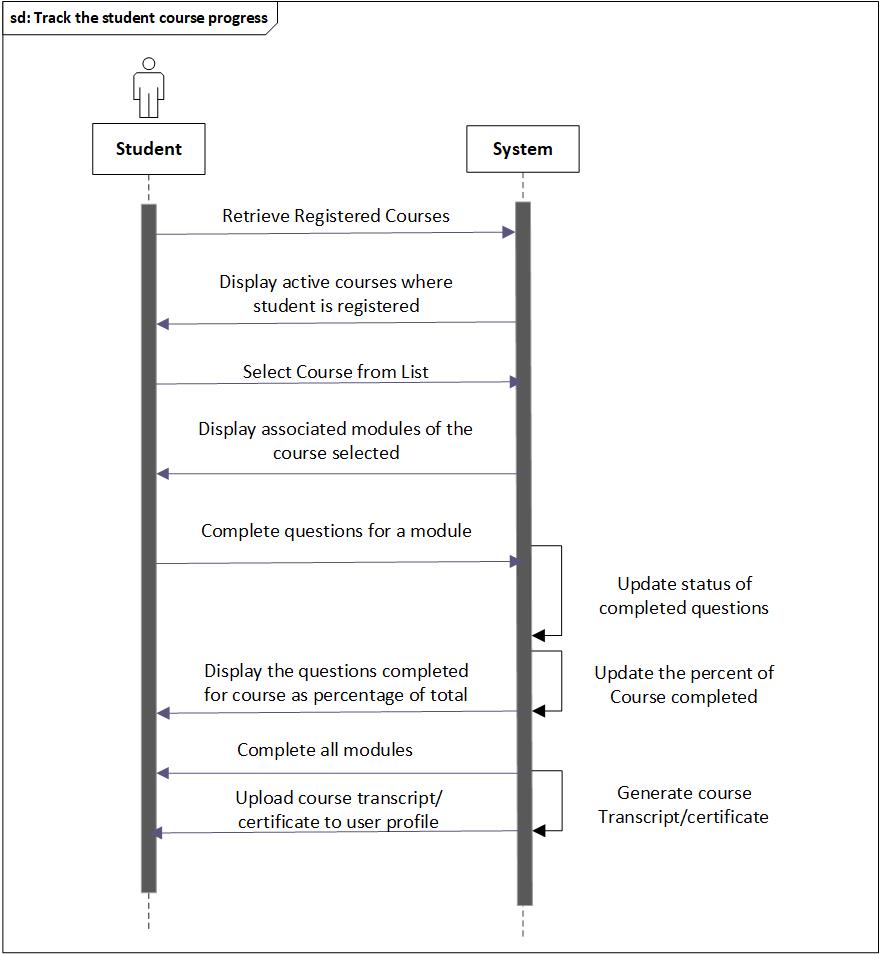


Figure 4: Sequence Diagram: Track the student course progress

# **Data Dictionary:**

There are five tables in the database schema: student, course, module, question, and instructor.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Datatype** | **Constraints** | **Description** |
| 1 | student\_id | int (12) | Primary key | Unique Id of user |
| 2 | student\_name | varchar (255) | - | Name of the user |
| 3 | Username | varchar (255) | Unique key | Login name of user |
| 4 | Password | varchar (255) | - | password of user |
| 5 | Email | varchar (255) | - | Email of user |
| 6 | phone\_number | varchar (255) | - | Phone number of the  User |
| 7 | registration\_date | Datetime | - | Date when student registered with CE shop |
| 8 | last\_login\_date | Datetime | - | Date of last login to CE shop account |

1. **Table Name: CE\_STUDENT**

**Table 1: CE\_STUDENT**

1. **Table Name: CE\_COURSE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Datatype** | **Constraints** | **Description** |
| 1 | course\_id | int (12) | Primary Key | Course Id |
| 2 | course\_student\_id | varchar (20) | Unique Key | Uniquely identifies course student pair |
| 3 | student\_id | int (12) | Foreign Key | References Student Table |
| 4 | course\_name | varchar (255) | - | Course name |
| 5 | Description | Text |  | Course description |
| 6 | instructor\_id | int (12) | Foreign Key | Course instructor |
| 7 | start\_date | Datetime | - | Date course started |
| 8 | end\_date | Datetime | - | Date course ended |
| 9 | archive\_status | Number | - | Archived: yes (0),  Active: no (1),  Initial: -1 |
| 10 | completion\_status | Number | - | All modules completed – (100%) |
| 11 | course\_fee | Number | - | Fee for taking course |
| 12 | Prerequisites | Text | - | Prerequisites for taking course |
| 13 | time\_to\_complete | Datetime | - | Total hours to complete course |
| 14 | maximum\_grade | varchar (2) | - | Maximum possible grade |

**Table 2: CE\_COURSE**

1. **Table Name: CE\_MODULE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Datatype** | **Constraints** | **Description** |
| 1 | Id | int (20) | Primary key | Unique module identifier |
| 2 | module\_id | varchar (10) | Primary key for current table and foreign key to course and student table | Module identifier |
|  |  |  |  |
| 3 | course\_id | int (20) | References Course table |
|  |  |  |  |
| 4 | student\_id | int (20) | References Student table |
|  |  |  |  |
| 5 | module\_name | varchar (255) | - | Name of module |
| 6 | Description | varchar (255) | - | Module description |
| 7 | total\_hours | Datetime | - | Total time to complete module |
| 8 | completion\_status | int (1) | - | All quizzes/lectures done – 1, Default – 0 |
| 9 | grade\_obtained | varchar (2) | - | Total grade of all the quizzes |

**Table 3: CE\_MODULE**

1. **Table Name: CE\_INSTRUCTOR**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Datatype** | **Constraints** | **Description** |
| 1 | instructor\_id | int (12) | Primary key | Unique Id of instructor |
| 2 | instructor\_name | varchar (255) | - | Name of the instructor |
| 3 | Username | varchar (255) | Unique key | Login name of user |
| 4 | Password | varchar (255) | - | password of user |
| 5 | Email | varchar (255) | - | Email of user |
| 6 | phone\_number | varchar (255) | - | Phone number of the  User |
| 7 | Licensed | int (1) | - | 0 – has license, 1 – no license |
| 8 | available\_hours | Datetime | - | Number of hours per week instructor is available |
| 9 | CDEI\_instructor | int (1) |  | 0 – no CDEI, 1- CDEI |
| 10 | years\_of\_exp | int (20) | - | Total teaching experience |
| 11 | additional\_details | int (12) | - | Additional details provided |

**Table 4: CE\_INSTRUCTOR**

1. **Table Name: CE\_QUESTION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.No** | **Field Name** | **Datatype** | **Constraints** | **Description** |
|  |  |  |  |  |
| 1 | question\_id | int (20) | Primary key | Question Id |
| 2 | module\_id | int (20) | Foreign key | References Module’s id field |
| 3 | total\_marks | int (20) | - | Total marks for assignment |
| 4 | marks\_obtained | int (20) | - | Marks obtained by student |
| 5 | description | Text | - | Question description |
| 6 | exam\_type | varchar(255) | - | Type of exam |
| 7 | question\_type | varchar(255) | - | Type of question |
| 8 | selected\_answer | varchar(10) | - | Answer selected by student |
| 9 | completion\_status | varchar(255) |  | Status of completion of question |

**Table 5: CE\_QUESTION**

# **IMPLEMENTATION**

# **Tools/Platforms & Technologies**

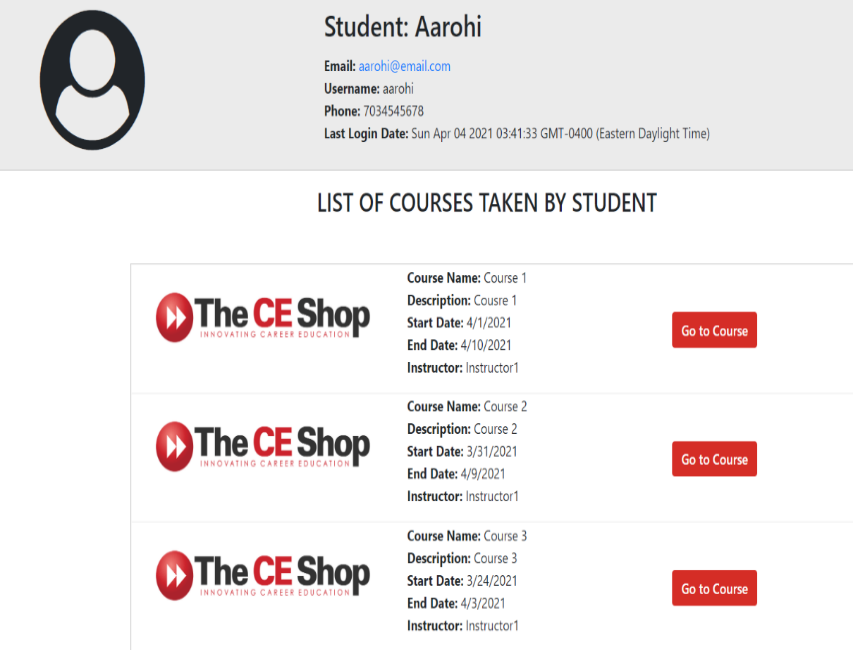
**Technologies used:**

1. **Back-end:** Java Spring MVC Framework and Spring Boot technology
2. **Front-end:** Angular, NodeJS, Bootstrap 4, and Typescript programming language.
3. **Database:** Amazon RDS Oracle Database instance

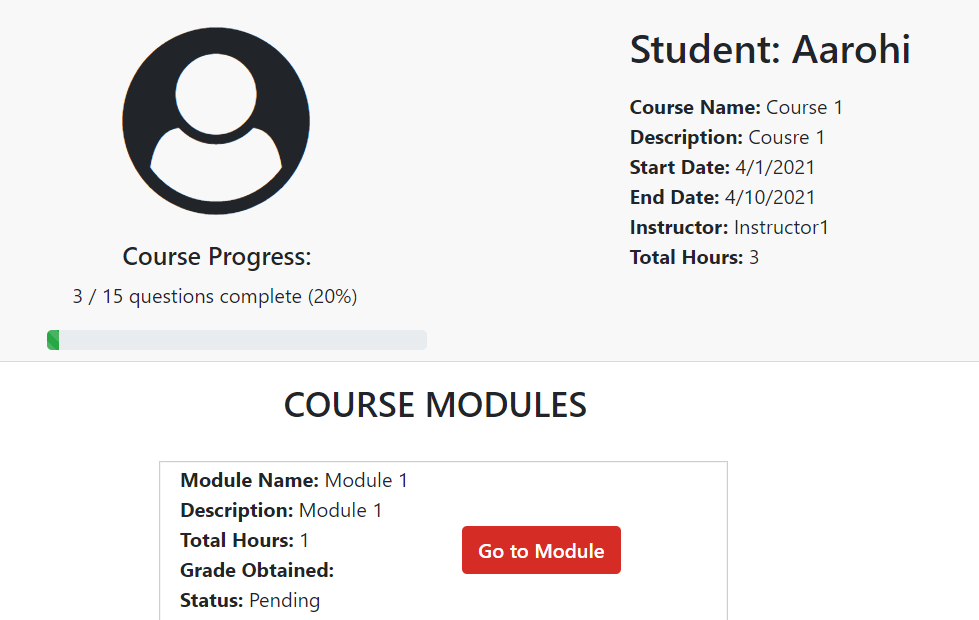
**Software tools used:**

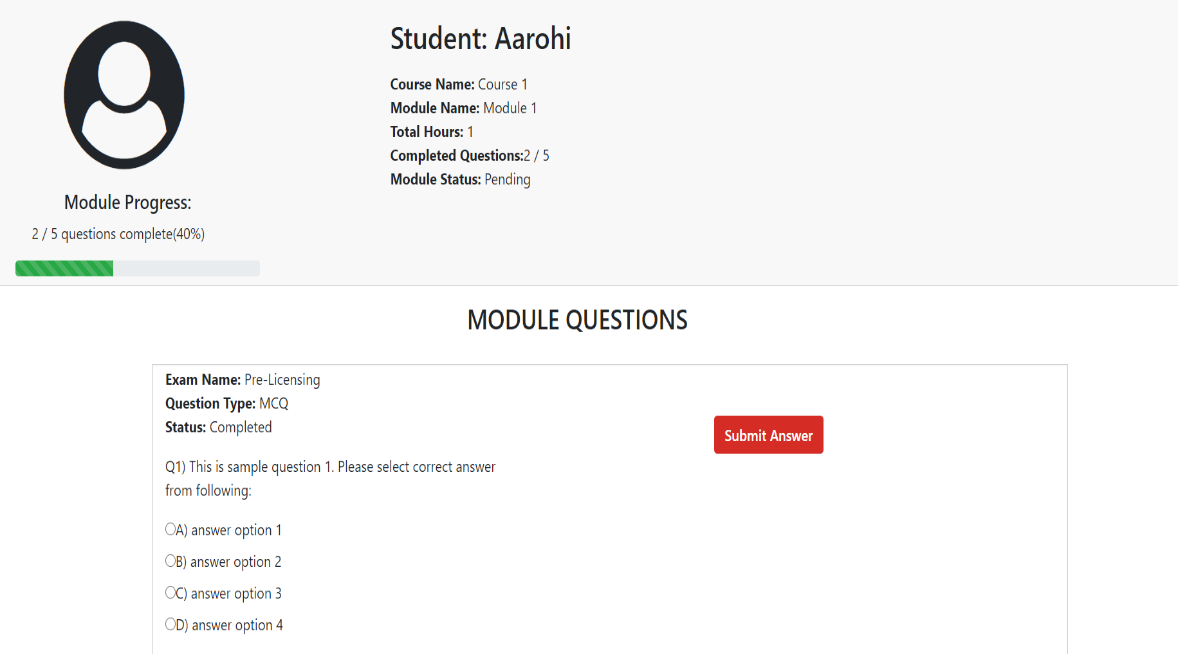
1. **Spring Boot** – for the backend Java code of application
2. **Microsoft Visual Studio Code** – for designing Angular forms and screens at the frontend
3. **Oracle SQL Data Developer** – for generating database schema
4. **Microsoft Visio** – for designing UML diagrams

# **User Interface Screenshots**



**Figure 5: Fetching the courses taken by student**





**Figure 6 (a) and 6(b): Monitoring the student p****rogress throughout course**

# **Code Implementation**

The following components are included in this project for creating an end-end solution of tracking the student progress throughout the course.

* + 1. **Student-Details:**

Fetches the list of courses taken by student by the course-Id. The component calls the student-service which forms the business layer that makes the HTTP GET request to the backend REST Controller implemented using Java Spring MVC framework. The data requests are made using Spring Hibernate ORM JPA repositories.

1. **Student-Details Component:** 
   * Fetches the list of courses taken by student using following code and navigates between components via angular routing.

|  |
| --- |
| this.studentService.getCourseByStudentId(this.studentId).subscribe(data =>{     this.courses = data;});  courseDetails(course,i){  sessionStorage.setItem('studentname',this.student.studentName);  sessionStorage.setItem('course',JSON.stringify(course));  this.router.navigate(['/coursedetails']); } |

1. **Student-Details Service:**
   * Retrieves the data from the Spring Controller via HTTP GET request in JSON format.

|  |
| --- |
| public getCourseByStudentId(studentId){   this.getStudentCourses = 'http://localhost:8080/student/get CourseByStudentId'+'/'+studentId;  return this.http.get<Course[]>(this.getStudentCourses); } |

1. **Java Spring StudentController:**
   * Receives the HTTP GET request from angular and returns the total count of questions across all modules.

|  |
| --- |
| @RequestMapping(value="/getCourseByStudentId/{id}",method=RequestMethod.GET)  @ResponseBody  public List<Course> getCourseByStudentId(@PathVariable("id") Long studentId){  return studentSer.getCourseByStudentId(studentId);} |

* + Receives the HTTP GET request from angular and returns the total count of completed questions across all modules.

|  |
| --- |
| @RequestMapping(value="/getCourseByStudentId/{id}",method=RequestMethod.GET)  @ResponseBody  public List<Course> getCourseByStudentId(@PathVariable("id") Long studentId){  return studentSer.getCourseByStudentId(studentId);} |

* + The business logic implemented in StudentService class with fetches list of courses corresponding to the student-id.

|  |
| --- |
| @Override  public List<Course> getCourseByStudentId(Long studentId)  {  return studentRepo.getCourseByStudentId(studentId);} |

1. **Database queries (StudentRepository):**
   * The database queries are executed using Hibernate query language of JPA repositories in spring. The following query fetches the list of courses that match the student-id.

|  |
| --- |
| @Query("FROM Course c where c.studentId= ?1")  List<Course> getCourseByStudentId(Long studentId); |

* + 1. **Course-Details:**

Fetches the modules based on course-id and counts the questions that are completed for each module. Next, it divides the completed modules (all questions completed) fetched by total questions for all the modules to get percentage of course completed.

1. **Course-Details Component:** 
   * Fetches the count of questions across all modules by module-id.

|  |
| --- |
| this.moduleService.getQuestionsCountByModuleId(this.modules.id).subscribe( data => {  sessionStorage.setItem(id,JSON.stringify(data));  this.tot\_questions += JSON.parse(sessionStorage.getItem(id));  sessionStorage.setItem("tot\_questions",JSON.stringify(this.tot\_questions));   }); |

* + Fetches the count of questions completed across all modules by module-id.

|  |
| --- |
| this.moduleService.getCompleteQuestionsCountByModuleId(this.modules[i].id).subscribe( data => {  sessionStorage.setItem(id,JSON.stringify(data));  this.complete\_questions += JSON.parse(sessionStorage.getItem(id));  sessionStorage.setItem("complete\_questions",JSON.stringify(this.complete\_questions)); }); |

* + Calculates the total percentage as follows:

|  |
| --- |
| this.completed\_course = (this.complete\_questions/this.tot\_questions)\*100; |

1. **Course-Details Service:**
   * Retrieves the count of total questions for course per module from the Spring Controller via HTTP GET request in JSON format.

|  |
| --- |
| public getQuestionsCountByModuleId(moduleId)  {this.getTotalQuestions = 'http://localhost:8080/module/getQuestionsCountByModuleId'+'/'+moduleId;   return this.http.get<number>(this.getTotalQuestions); } |

* + Retrieves the count of total completed questions for course per module from the Spring Controller via HTTP GET request in JSON format.

|  |
| --- |
| public getCompleteQuestionsCountByModuleId(moduleId)  {this.getCompleteQuestions = 'http://localhost:8080/module/getCompleteQuestionsCountByModuleId'+'/'+moduleId;  return this.http.get<number>(this.getCompleteQuestions);} |

1. **Java Spring CourseController:**
   * Receives the REST request from angular and returns the count of total questions.

|  |
| --- |
| @RequestMapping(value="/getQuestionsCountByModuleId/{id}",method=RequestMethod.GET)  @ResponseBody  public Integer getQuestionsCountByModuleId(@PathVariable("id") Long moduleId)  {return moduleSer.getQuestionsCountByModuleId(moduleId);} |

* + Get total completed question count for current module from database.

|  |
| --- |
| @RequestMapping(value="/getCompleteQuestionsCountByModuleId/{id}",method=RequestMethod.GET)  @ResponseBody  public Integer getCompleteQuestionsCountByModuleId(@PathVariable("id") Long moduleId) {return moduleSer.getCompleteQuestionsCountByModuleId(moduleId)} |

1. **Database queries:**
   * The database queries are executed using Hibernate query language of JPA repositories in spring. The following query fetches total questions per module.

|  |
| --- |
| @Query("select count(q) from Question q where q.module.id= ?1")  Integer getQuestionsCountByModuleId(Long moduleId); |

* + This query fetches the list of completed questions that match the module-id.

|  |
| --- |
| @Query("select count(q) from Question q where q.module.id= ?1 and q.completionStatus=1")  Integer getCompleteQuestionsCountByModuleId(Long moduleId); |

* + 1. **Questions-Details:**

Updates the count-of questions submitted by the user and modifies course completed progress accordingly.

1. **Question-Details Component:** 
   * Updates the completed Status of questions completed.

|  |
| --- |
| this.questionService.updateQuestion(question).subscribe( data =>  {console.log("Updated data successfully:"+data);}); |

1. **Student-Details Service:**
   * Posts the data from the Spring Controller via HTTP POST request in JSON format and the service updates the status of question recieved.

|  |
| --- |
| public updateQuestion(question)  {return this.http.post<Question>(this.updtQues, question);} |

1. **Java Spring StudentController:** 
   * Receives the POST request from angular updates the status of completed questions.

|  |
| --- |
| @CrossOrigin(origins = {"http://localhost:4200"},allowedHeaders="\*")  @RequestMapping(value="/updateQuestion",method=RequestMethod.POST)  //@ResponseBody  public @ResponseBody String updateQuestion(@RequestBody Question question) { String answer = question.getSelectedAnswer();  question.setSelectedAnswer(answer);  int completedStatus=1;  question.setCompletionStatus(completedStatus);  try { questionRepo.save(question); }  catch (Exception ex)  { return "Error creating the project: " + ex.toString();}  return "Question updated created! (id = "+question.getQuestionId()+ ")";} |

# **RECOMMENDATIONS**

As a part of enhancement to the current project, the system can be deployed on the cloud using Amazon Web Services, and automated code can be written using AWS Lambda to update the student’s course progress in real-time based on the amount of course completed by the student. This code will periodically check the number of modules finished by the student and update the course progress accordingly. Apart from this Elastic Stack can be used for helping users in searching the courses and analyzing the progress made by the students in a particular course, by analyzing the log data on the server related to the course using Logstash/beats and display it using data visualization dashboards of Kibana.

Additionally, big data and machine learning technologies can be used to give future course recommendations to the students based on their course preferences. Graph databases such as Neo4j can be used to generate the recommendation matrix for such type of system. The system can also track whether the student will be able to complete the course/not, based on the progress made by the student currently. Also, the system can send timely notifications to users, based on their progress to complete the assignments and lectures before the due date, thereby enabling them to successfully complete the courses.