Faculty of Computers & Informatics

Benha University

E-PEAK

(Distance Learning System)

Presented by :

1. Abanoub Fayez Kamal Mikhail .
2. Ahmed Fathy Abd-Elfattah Hassan.
3. Manar Elsayed Mohamed Ezzat.
4. Hala Mahmoud Gouda Elsayed.
5. Yara Abd El Rahman Mahmoud Ali.
6. Yousef Reda Saed El-Hadad.

Under Supervision of

Dr. Ahmed Taha

Declaration

We hereby certify that this material, which we now submit for assessment on the program of study leading to the award of Bachelor of Computers and Informatics in computer science is entirely our own work, that we have exercised reasonable care to ensure that the work is original, and does not to the best of our knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of our work.

Signed: _____

_____

_____

_____

_____

_____

Date: Thursday, 4 July 2019.

Abstract

As an abstract for this document, this document has been prepared by the College of computer and informatics, Benha University. In this document ,we will describe all phases that lead us to complete this project.

About the project:-

E-peak is a web application which is useful for students who are going to choose any course in higher education, for them this application is helpful to make the correct the decision with the accurate information. In addition, E-peak is useful also for Instructors to manage their courses easily.

Table of content

CONTENTS

1.INTRODUCTION 7

PROBLEM STATEMENT 7

CURRENT SOLUTIONS 7

SOFTWARE DEVELOPMENT CYCLE 8

REQUIREMENT ANALYSIS 8

DESIGN 8

IMPLEMENTATION 8

QUALITY ASSURANCE 8

TESTING & INTEGRATION 8

MAINTENANCE 8

2. PLANNING AND SYSTEM REQUIREMENT 9

PLANNING OF THE SYSTEM. 9

TIME ESTIMATION 9

USER REQUIREMENTS 11

FUNCTIONAL REQUIREMENTS 11

NON FUNCTIONAL REQUIREMENTS 26

SYSTEM STACKHOLDERS 26

CONTEXT DIGRAM 27

DATA FLOW DIAGRAM 28

ENTITY RELATIONSHIP DIAGRAM (ERD) 29

3. SOFTWARE AND HARDWARE PLATFORMS 31

METHODOLOGY: - 31

EPICS: 31

STORIES TABLES: 31

TOOLS 33

INTELLIJ IDEA: 33

MYSQL WORKBENCH 33

TORTOISESVN : 33
NODE JS 34
THE JAVA DEVELOPMENT KIT (JDK) 34
VERSION CONTROL: 35
TECHNOLOGIES 36
COMPONENTS 37
STRUCTURAL DIRECTIVES 37
ROUTING 37
TYPESCRIPT (TS) 38
SPRING BOOT 38
SPRING JDBC 38
RESTFUL WEB SERVICES (API) 39
HTML5 40
CSS3 40
PROJECT ARCHITECTURE 41
FRONT-END LAYER 42
BACK-END LAYER 42
DATATIER LAYER 44
SUMMARY OF ARCHITECTURE: 44
IMPLEMENTAION DETAILS 45
COURSE RES 45
QUIZ RES 53
ATTANDANCE RES 57
GRADE RES 57
USER RES 58
LECTURE RES 59
ATTACHMENT RES 59
DEPLOY PROJECT USING MAVEN 61
REFERENCES 62

List of figures:-

LIST OF FIGURES

Figure 1 Learning Systems 7
Figure 2 Software Development Cycle 8
Figure 3 Gantt chart 10
Figure 4 Milestone 10
Figure 5 Use Case 11
Figure 6 Context diagram figure 27
Figure 7 user management 28
Figure 8 ERD Diagram 30
Figure 9 Intellij 33
Figure 10 my SQL workbench 33
Figure 11 Tortoize Svn 33
Figure 12 nodejs 34
Figure 13CLI 34
Figure 14 JDK 35
Figure 15 history 35
Figure 16 angular 36
Figure 17 type script 38
Figure 18 spring boot 38
Figure 19 Rest API 39
Figure 20 Html&CSS 40
Figure 21 project architecture 41
Figure 22 authentication 43
Figure 23 authorization 44
Figure 24 maven project 61
Figure 25 project after deployment 61

Chapter One

1. INTRODUCTION

Technology is changing the way teacher and learner interactions. Earlier learning has only one option that is to go to school and learn, but today we have various options of learning whatever we want. Some of the modern options of learning include e-Learning, online learning, distance learning, blended learning, digital learning and virtual learning along with class room learning.

Today's learners want relevant, mobile, self-paced, and personnel content. This need is fulfilled with the online learning here, students can learn at their own comfort and with their own requirement.

The online method of learning is the best suited for everyone. This digital revolution has led to remarkable changes in how the content is accessed, consumed, discussed, and shared. Online educational courses can be taken up by office goers and housewives too at the time that suits them. Depending on their availability and comfort, many people choose to learn at weekends or evenings.

Course management system CMS is a collection of software tools providing an online environment for course interactions. A CMS typically includes a variety of online tools and environments. It is used very widely in universities and schools.

Figure 1 Learning Systems

PROBLEM STATEMENT

With the advent in technology and with the perpetual increase in the strength of the students and the number of departments in the educational institutions, it is laborious to exchange the study materials between students and faculties members.

In most schools and universities, different teachers and professors overwhelm students with many education sites where they upload the resources of their subjects. Student finds it hard to look for materials, or announcements.

The main objective is to help the students get over the traditional methods of learning and use the internet where the notes for their respective subjects are easily available. It provides an automation procedure of studying online. The implementation of this project helps both the students and the teachers. The teachers can upload their materials on to the website by using their unique ID and the students can gain access to these materials.

CURRENT SOLUTIONS

Social media sites are used excessively for this purpose for communications between students and their teachers, also different storage services are used for uploading materials (ex: Google Drive).

SOFTWARE DEVELOPMENT CYCLE

REQUIREMENT ANALYSIS

In this stage we have searched in the Learning management systems field trying to find a way to help in improving education process, we get some information from professors we met, the team also make a survey for what difficulties students face while studying and what is needed to facilitate the educational process. This information is then used to plan the basic project approach and creating several scenarios, planning for the quality assurance, learning about new up to date technologies in web development.

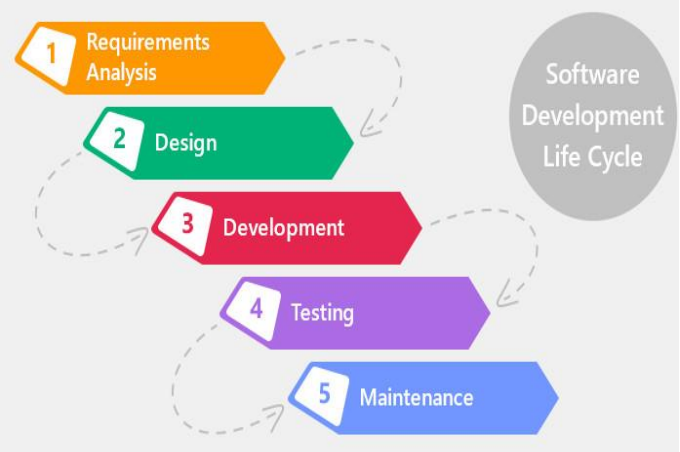


Figure 2 Software Development Cycle

DESIGN

In this stage the team produce more than one designs of the software based on the scenarios we have in the previous stage, then chooses the perfect design.

IMPLEMENTATION

In this stage of SDLC the actual development starts and the product is built. The programmers build prototype using Java as a programming language and Spring framework as the backend, and Angular as the front-end.

The prototype is iterated over for enhancement until satisfaction. Programming team has divided the system into modules working on them in parallel to complete the implementation stage in time.

QUALITY ASSURANCE

By following the studied steps, splitting tasks on the team, estimating time and resources required to complete each task, ensuring that software is delivered on time, on schedule and in accordance with the requirements of system.

The project plan is settled and resources are available and work broke down.

TESTING & INTEGRATION

In this stage we firstly checked that is all modules error free and perform what's they designed for and then test the application as a whole to make sure that all is doing well.
The application may first be released in a limited segment and tested in the by some users to get their feedbacks, based on them the product may be released as it is or not.

MAINTENANCE

Improve performance or other attributes based on the feedbacks and fixing any bugs appeared in the testing stage for the next updates

Chapter Two

2. PLANNING AND SYSTEM REQUIREMENT

The purpose of this section provides details about the system functionality. It also introduces stockholders and their interaction with the system. It mentions the system assumptions about the solution. It provides scope of the system, future work and time planning. Further, it also provides the requirements specifications in detailed terms and a description of the different system interfaces. Different specification techniques are used in order to specify the requirements more precisely for different audiences.

PLANNING OF THE SYSTEM.

As part of the implementation, we need to implement system that helps students, Instructors and make educational activities online. Each one of them has specific services to use.

- **IN SCOPE**

  1. User Management.
  2. Course Management.
  3. Administrations.

- **FUTURE WORK**
  1. Localization
  2. Notification
  3. Chat

TIME ESTIMATION

Accurate time estimation is a skill essential for good project management. It is important to get time estimates right for two main reasons:

1. Time estimates drive the setting of deadlines for delivery and planning of projects, and hence will impact on other people assessment of your reliability and competence as a project manager.

2. Time estimates often determine the pricing of contracts and hence the profitability of the contract/project in commercial terms.

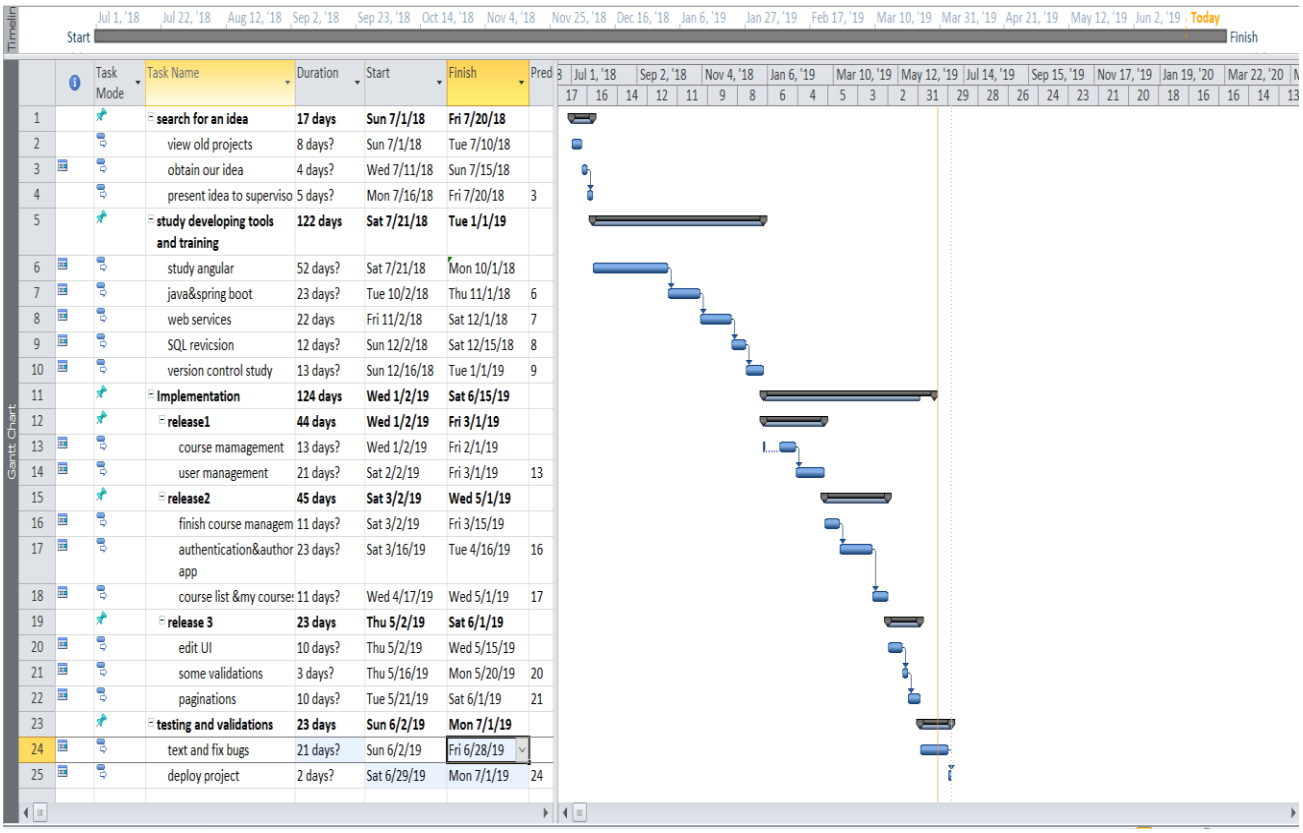


Figure 3 Gantt chart

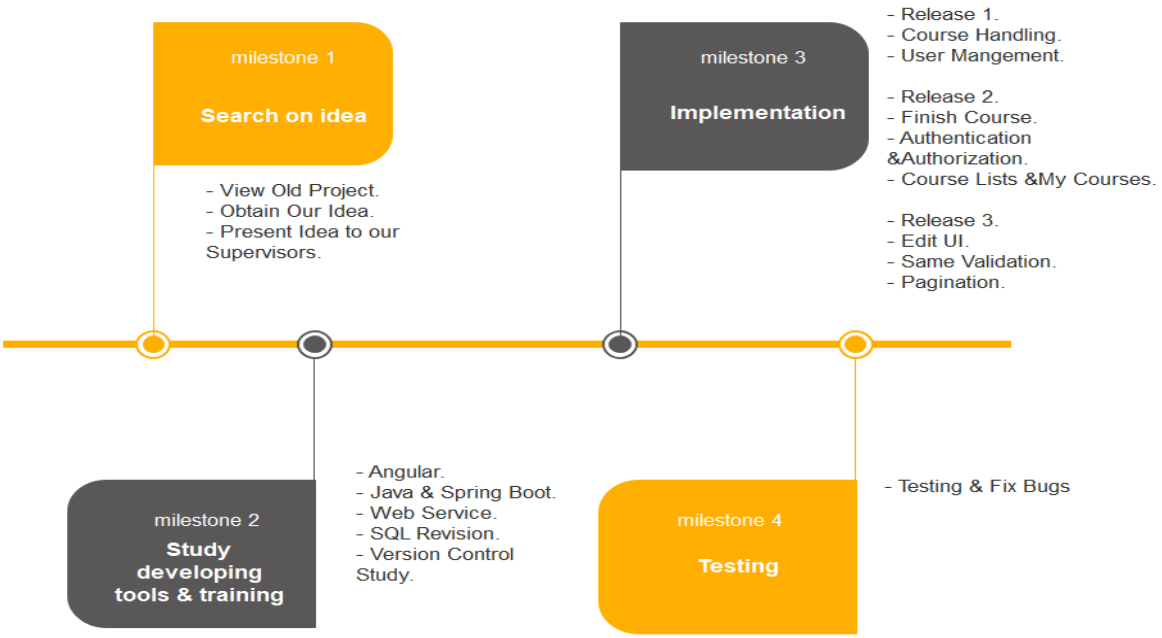


Figure 4 Milestone

USER REQUIREMENTS

This section includes requirements that directly affect the customer. It describes the functional and non–functional requirements of the system. The section also describes other specifications related to the user requirements.

1. FUNCTIONAL REQUIREMENTS

The use cases diagrams below illustrate the system's functional requirements

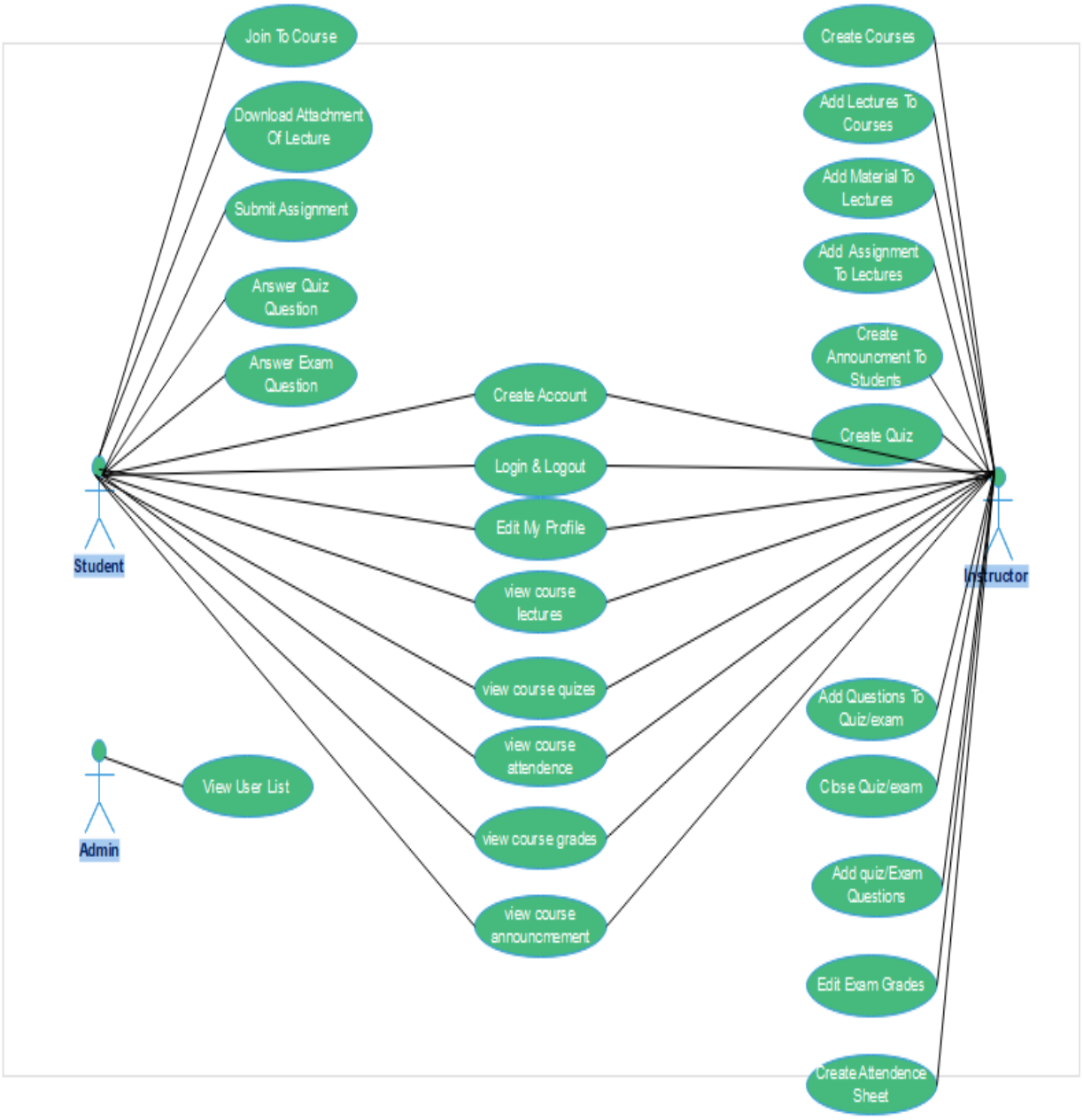


Figure 5 Use Case

WELCOME USE CASE

Use Case Name	Welcome Page
Use case number	1
Users	Any User (Visitor or System User)
Description	This use case is used to navigate general user to the features of system and navigate him to Login and Signup
Assumptions	None
Pre-Conditions	None
Basic Flow	1. User open web application and see welcome page and features of system also information about system. 2. Then, he will decide if he will Sign up, login or quit.
Post-Conditions	The system navigates the user to the login page.
Blocking Flow	None.

USER SIGNUP USE CASE

Use Case Name	student signup use case
Use case number	2
Users	Any User (Visitor or System User)
Description	This use case is used to help user to create account as a student or Instructor.
Assumptions	None.
Pre-Conditions	Actor is a system user.
Basic Flow	1- User fill sign up form that include username, password-mail …etc. 2- Then, press register.
Post-Conditions	The system navigates the student to login to the new account User's LOGIN USE CASE
Blocking Flow	None.

USER'S LOGIN USE CASE

Use Case Name	user's login use case
Use case number	3
Users	Admin, System User
Description	This use case is used to login user into the application.
Assumptions	None
Pre-Conditions	Actor is a System user.
Basic Flow	1. User enters username & password. 2. User presses the login button.
Post-Conditions	The user has been authenticated and authorized to use the account then ,The system navigates the user to the course list page.
Blocking Flow	The user enters invalid username and/or password → *the system shows an error message.*

USER'S LOGOUT USE CASE

Use Case Name	user's logout use case
Use case number	4
Users	Admin ,system User
Description	This use case is used to logout user from the application.
Assumptions	Actor is already signed in.
Pre-Conditions	None.
Basic Flow	1. The user clicks the logout link/button.
Post-Conditions	The user is logged out from the application. The system navigates the user to the login page.

VIEW USER PROFILES USE CASE

Use Case Name	Load user profiles use case
Use case number	5
Users	Admin
Description	This use case is used to load users' profiles page.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is authorized to view users' privileges.
Basic Flow	1. The user selects to open/view the Profiles page. 2. The system loads the Profiles entries from the database.
Post-Conditions	The system navigates the user to the profiles page. The system loads the users' profiles.
Blocking Flow	None.
Alternative Flow	None.

EDIT USERS PROFILES USE CASE

Use Case Name	edit users profile use case
Use case number	6
Users	system user
Description	This use case is used to edit and save users' profiles.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is authorized to view users' privileges.
Basic Flow	1. User navigates to the users' profiles page using use case 2. User applies his changes. 3. User clicks the save button.
Post-Conditions	The changes have been updated in database. Then , system shows that the user changes have been done successfully.

ALL COURSES USE CASE

Use Case Name	all courses use case
Use case number	7
Primary Actor	Student, Instructor
Description	This use case is used to load all courses in the system.
Assumptions	Actor is already signed in.
Pre-Conditions	None
Basic Flow	1- User click courses button.
Post-Conditions	The system loads other all courses.
Blocking Flow	None

USER COURSES USE CASE

Use Case Name	user courses use case
Use case number	8
Primary Actor	Student ,Instructor
Description	This use case is used to load all courses that student is in or created by instructor.
Assumptions	Actor is already signed in.
Pre-Conditions	None
Basic Flow	**The following is the basic flow of this use case:** 2- User click my courses button.
Post-Conditions	The system loads other all courses that student is in or created by instructor.
Blocking Flow	None

CREATE NEW COURSE

Use Case Name	create new course use case
Use case number	9
Primary Actor	Instructor.
Description	This use case is used to create new course.
Assumptions	Actor is already signed in.
Basic Flow	1- User clicks on create course button. 2- Then, user fill in the form and press create button.
Post-Conditions	The system navigates to course dashboard.

JOIN NEW COURSE

Use Case Name	join new course use case
Use case number	10
Primary Actor	Student
Description	This use case is used to join new course.
Assumptions	Actor is already signed in.
Pre-Conditions	None
Basic Flow	1- User clicks on course name. 2- Then, system will load course information with enroll button. 3- Student clicks on enroll if he isn't enrolled before.
Post-Conditions	The system navigate to course lectures pages.

VIEW ANNOUNCEMENT

Use Case Name	view announcement use case
Use case number	11
Primary Actor	Student, Instructor
Description	This use case is used to view announcement of the course
Assumptions	Actor is already signed in. Actor is the instructor of the course or student in the course.
Pre-Conditions	None
Basic Flow	1- User clicks on announcement.
Post-Conditions	system will load list of course announcement

CREATE ANNOUNCEMENT

Use Case Name	create announcement use case
Use case number	12
Primary Actor	Instructor
Description	This use case is used to add new announcement to students of the course.
Assumptions	Actor is already signed in. Actor is the instructor of the course.
Pre-Conditions	None
Basic Flow	1- User click on create announcement button. 2- User write announcement and click share
Post-Conditions	system will add the new announcement and navigate to view announcements page

VIEW COURSE LECTURES

Use Case Name	view lectures use case
Use case number	13
Primary Actor	Instructor, student
Description	This use case is used to view lectures of the course.
Assumptions	Actor is already signed in. Actor is the instructor of the course or student in the course.
Pre-Conditions	None
Basic Flow	1- User click on lectures button.
Post-Conditions	System will course lectures page.

CRETAE LECTURE

use case name	view lectures use case
Use case number	14
Primary Actor	Instructor
Description	This use case is used to add new lecture to the course.
Assumptions	Actor is already signed in. Actor is the instructor of the course or student in the course.
Pre-Conditions	None
Basic Flow	1- User click on create lecture button. 2- Fill the form of create lecture. 3- Press create button
Post-Conditions	System will add the new lecture and navigate user to course lectures page.

ADD ATTACHMENT TO LECTURE

Use Case Name	materials use case
Use case number	15
Primary Actor	Instructor
Description	This use case is used to enable instructor to upload materials of the course.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is creator of the course.
Basic Flow	1. User click add material button. 2. Then, select files to upload.
Post-Conditions	System upload files and navigate user to lecture details page
Blocking Flow	None

DOWNLOAD ATTACHMENT

Use Case Name	Download attachment use case.
Use case number	16
Primary Actor	student
Description	This use case is used to view and download materials of the course.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is enrolled in the course.
Basic Flow	1- User click material button. 2- System will load lecture attachment. 3- Student click on file name.
Post-Conditions	The file will be downloaded on user PC.
Blocking Flow	None

SUBMIT ASSIGNMENTS

Use Case Name	assignments use case
Use case number	17
Primary Actor	Student
Description	This use case is used to submit assignments of the course before deadline.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is enrolled in any course.
Basic Flow	1- User click upload button. 2- Select folder from computer. 3- Click submit
Post-Conditions	The system send file to the admin of the course (instructor).
Blocking Flow	If the student submit assignment after deadline show error message.

VIEW COURSE QUIZES/EXAMS USE CASE

Use Case Name	view exams use case
Use case number	18
Primary Actor	Instructor, student
Description	This use case is used to view lectures of the course.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is the instructor of the course or student in the course
Basic Flow	1- User click on quizzes/exams button.
Post-Conditions	System will load course quizzes/exams page.

CREATE QUIZ

Use Case Name	create quiz use case
Use case number	19
Primary Actor	Instructor
Description	This use case is used to add quiz to the course.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is the instructor of the course.
Basic Flow	1- User click on create quiz button. 2- User fill the form and press next button.
Post-Conditions	system will add the new quiz and navigate to add quiz/ page exam questions page

ADD QUIZ/EXAM QUESTIONS

Use Case Name	create quiz use case
Use case number	20
Primary Actor	Instructor
Description	This use case is used to add questions to the quiz/exam.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is the instructor of the course.
Basic Flow	1- User select question type. 2- Then add questions to quiz 3- Click submit button.
Post-Conditions	System will add the questions and navigate user to course quizzes/exams.

ANSWER QUIZ AND EXAM QUESTIONS

Use Case Name	Answer questions use case.
Use case number	21
Primary Actor	Student
Description	This use case is used to answer questions of quizzes and exams.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is enrolled in any course.
Basic Flow	1- User click on quiz/exam name. 2- If he doesn't answer before the system will load quiz questions page.
Post-Conditions	The system navigates to course quizzes/ course exams page
Blocking Flow	Actor is not enrolled in any course.

VIEW QUIZ/EXAM MAIN DETAILS USE CASE

Use Case Name	View quiz/exam main details use case.
Use case number	22
Primary Actor	Instructor
Description	This use case is used to view quiz/exam main details.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is creator of the course.
Basic Flow	User click on quiz/exam name.
Post-Conditions	If he is the instructor of the course, the system will load main details of the quiz/exam.
Blocking Flow	Actor is not creator of the course.

VIEW QUIZ/EXAM RESULT USE CASE

Use Case Name	View quiz/exam result use case.
Use case number	23
Primary Actor	Student.
Description	This use case is used to view quiz/exam result.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is student in the course.
Basic Flow	1- User click on quiz/exam name.
Post-Conditions	If he is the instructor of the course, the system will load main details of the quiz/exam.
Blocking Flow	Actor is not student in the course.

VIEW GRADES USE CASE

Use Case Name	View grades of the exams use case.
Use case number	24
Primary Actor	Student, instructor
Description	This use case is used to view grades of the exams.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is student in the course or instructor of the course.
Basic Flow	1- User click on grades button.
Post-Conditions	If he is the instructor of the course, the system will load the grades of all students. If he is student in the course, the system will load his grades only
Blocking Flow	Actor is not student in the course or not creator of the course.

EDIT GRADES USE CASE

Use Case Name	Edit grades of the exams use case.
Use case number	25
Primary Actor	instructor
Description	This use case is used to edit grades of the exams.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is the instructor of the course.
Basic Flow	1- User click on edit grades button.
Post-Conditions	The system loads the grades of all students enabled for editing.
Blocking Flow	Actor is not creator of the course.

VIEW ATTENDENCE USE CASE

Use Case Name	View attendance of the course.
Use case number	26
Primary Actor	instructor, student
Description	This use case is used view attendance of the course.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is the instructor of the course or student of the course.
Basic Flow	1. User click on attendance button.
Post-Conditions	If he is the instructor of the course, the system will load attendance of all students. If he is student in the course the system will load his attendance only.
Blocking Flow	Actor is not creator of the course or student in the course.

CREATE ATTENDECE SHEET USE CASE

Use Case Name	Create attendance of the course.
Use case number	27
Primary Actor	instructor
Description	This use case is used create attendance of the course.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is the instructor of the course.
Basic Flow	1- User click on create attendance button.
Post-Conditions	The system loads/creates attendance sheet page.
Blocking Flow	Actor is not creator of the course or student in the course.

VIEW USER LIST USE CASE

Use Case Name	View attendance use case
Use case number	28
Primary Actor	System admin.
Description	This use case is used to view list of all users in the system.
Assumptions	Actor is already signed in.
Pre-Conditions	Actor is having a role of system admin.
Basic Flow	1- User click users' button.
Post-Conditions	The system loads updated attendance file after every lecture.

Non Functional Requirements

Here we specify some nonfunctional constraints that the program satisfies in order to be more concrete and stable.

- **Operational.**

  Our system is designed for Personal Computers, Mobile & Tablets, it only needs a valid connection to the Internet and web browser.

- **Performance requirement.**

  1. The system operates in real-time with high speed and responsive time under any circumstances.
  2. If unable to process HTTP request, should show error message.
  3. Web pages are loaded in a few seconds.
  4. HTTP request should be performed very quickly.
  5. Design should be attractive.

- **Security requirements**

  Users should be authenticated.

  1. User should have privilege to access specific pages and takes specific actions.
  2. The details need to be maintained properly.

- **Maintainability: -**

  Developers of the applications always reads the users' feedback on the system and respond fast to any bugs or problems.

SYSTEM STACKHOLDERS

Role	Description
Instructors	They used the system to manage their own courses
Students	They used the system to reach to contests of their courses.

CONTEXT DIGRAM

The Context Diagram shows the system under consideration as a single high-level process and then shows the relationship that the system has with other external entities (systems, organizational groups, external data stores, etc.).

- All external entities are shown on the context diagram as well as major data flow to and from them.
- The diagram does not contain any data storage.
- The single process in the context-level diagram, representing the entire system, can be exploded to include the major processes of the system.
- It also called level 0 data flow diagram.

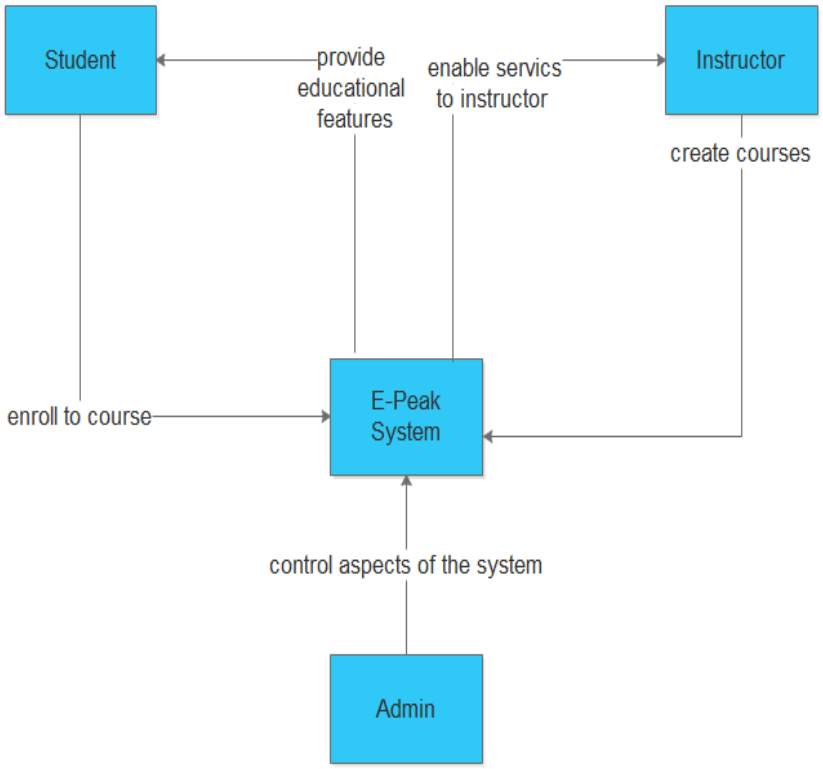


Figure 6 Context diagram figure

DATA FLOW DIAGRAM

"A data flow diagram, or a DFD, is a visual representation of any process or system's flow of information. By mapping out your process or system's flow of data, DFDs help you better understand your process or system, uncover its kinks, improve it, and can even help you implement a new process or system. DFDs can range from simple overviews to complex, granular displays of a process or system." hubspot.coma

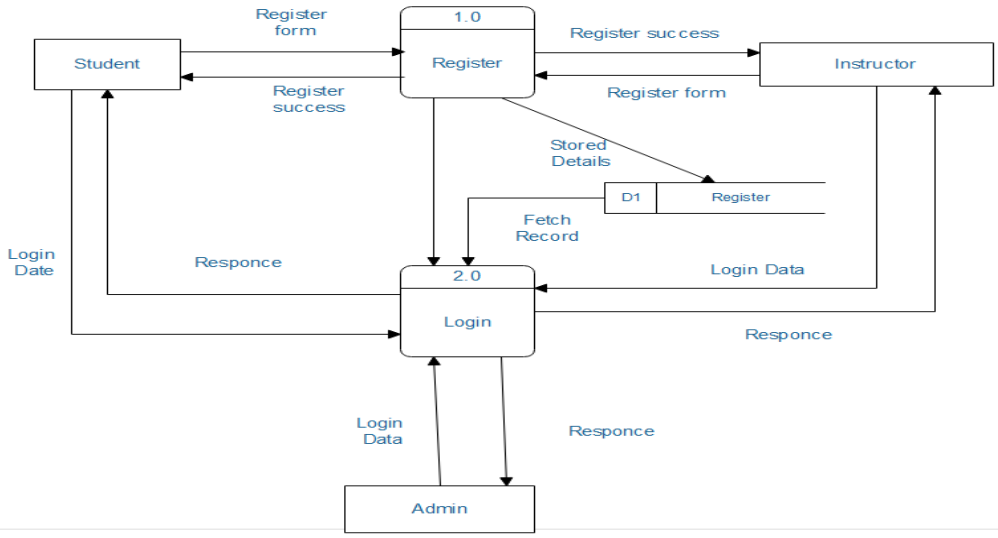


Figure 7 user management

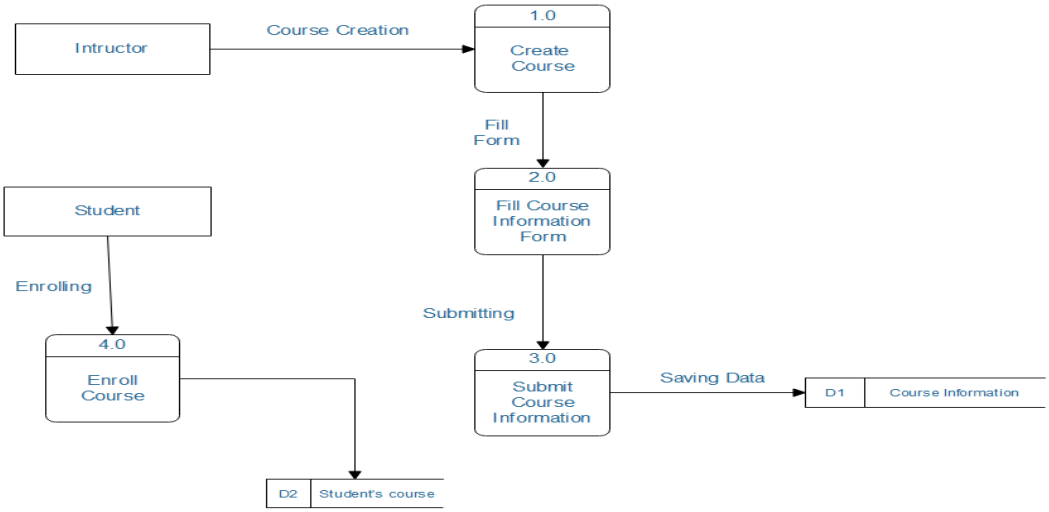
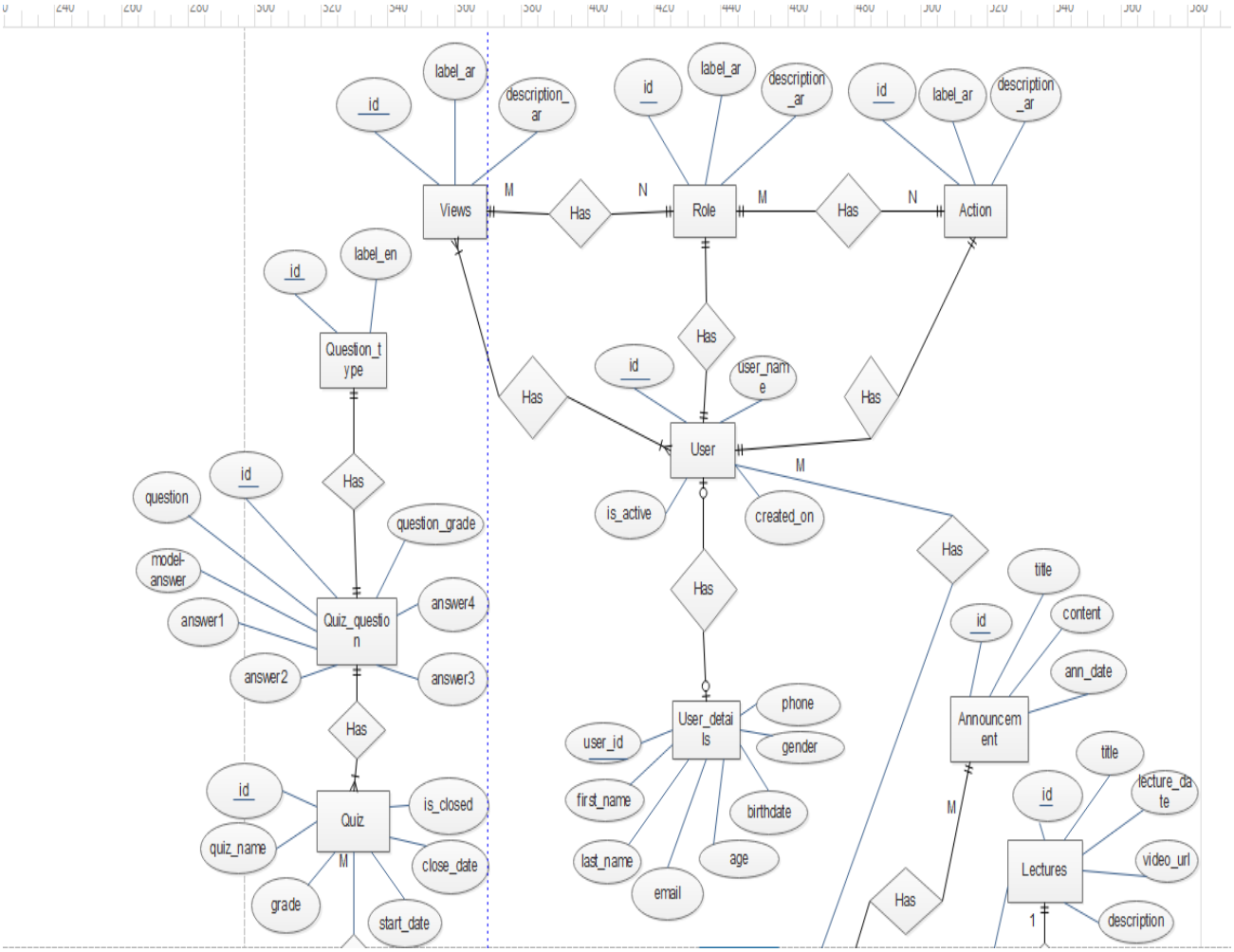


Figure 8 create course DFD

ENTITY RELATIONSHIP DIAGRAM (ERD)

Entity relationship diagram displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases. At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique.

- **ERD Benefits.**
  1. ER model allows you to draw Database Design.
  2. It is an easy to use graphical tool for modeling data
  3. Widely used in Database Design
  4. It is a GUI representation of the logical structure of a Database
  5. It helps you to identify the entities which exist in a system and the relationships between those entities.



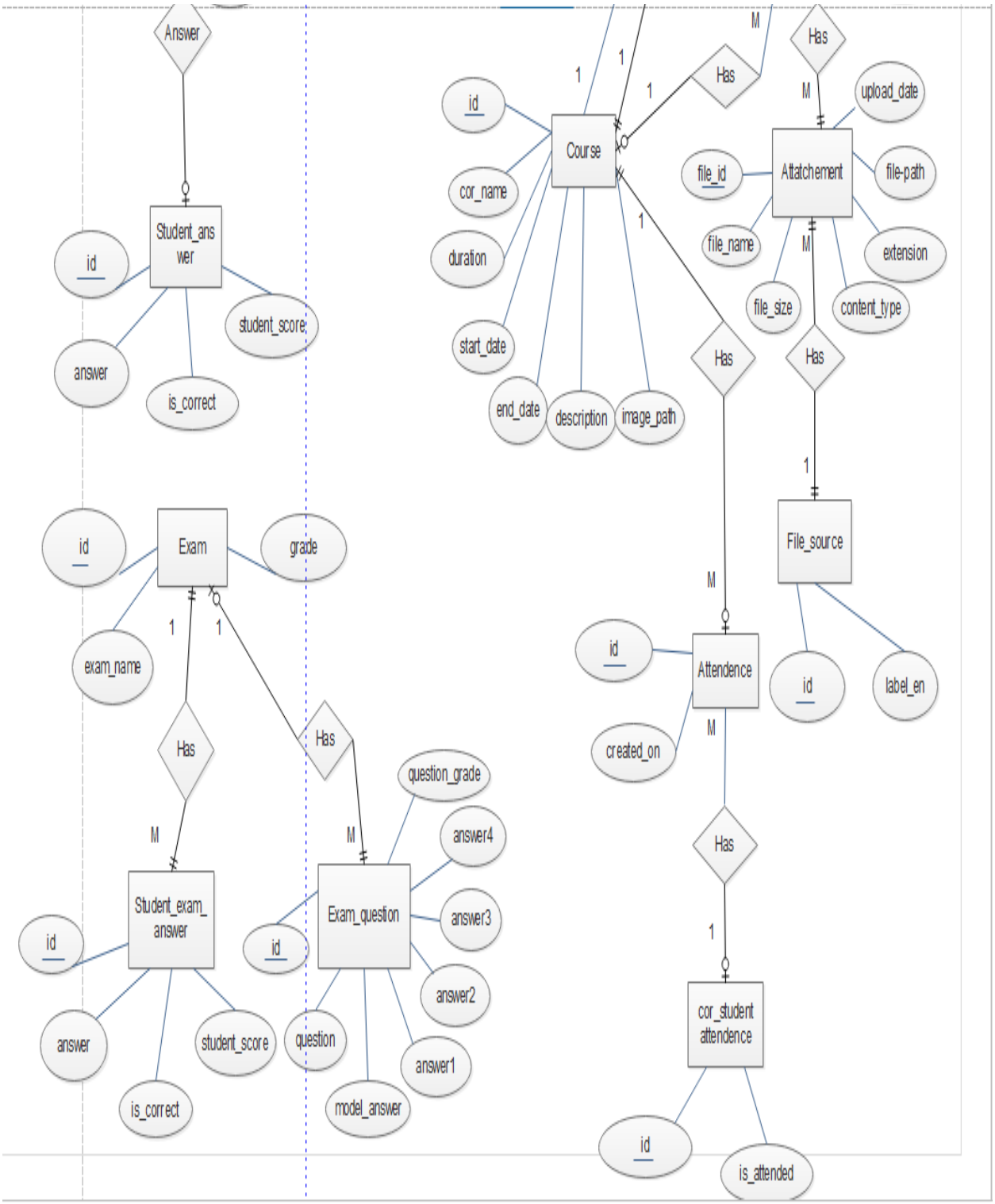


Figure 8 ERD Diagram

Chapter Three

3. SOFTWARE AND HARDWARE PLATFORMS

METHODOLOGY: -

We used Agile Methodology which is a process by which a team can manage a project by breaking it up into several stages (Epics & Stories) and involving constant collaboration with stakeholders and continuous improvement and iteration (Sprints) at every stage.

Agile methodology begins with clients describing how the end product will be used and what problem it will solve. This clarifies the customer's expectations to the project team.

Then the project passes through a process of planning, executing, and evaluating which might just change the final deliverable to fit the customer's needs. Continuous collaboration is a key among both team members and project stakeholders, to make fully-informed decisions.

First Project breaking down is into Epics.

EPICS:

1. Course List
2. Course Details
3. Security
4. Administration

STORIES TABLES:

Story Name	Epic	Assignee
[EP-1] Register	security	Ahmed Fathy
[EP-2] Login	security	Ahmed Fathy
[EP-3] User Profile	security	Manar El Sayed
[EP-4] Edit USER	security	Manar El Sayed
[EP-4] User List	administration	Manar El Sayed
[EP-5] My Courses	Course List	Hala Mahmoud
[EP-6] All Courses	Course List	Hala Mahmoud

[EP-7] View exam grades for Instructor	Course details	Hala Mahmoud
[EP-8] View exam grades for Student	Course details	Hala Mahmoud
[EP-9] View quiz grades for Instructor	Course details	Hala Mahmoud
[EP-10] View quiz grades for Student	Course details	Hala Mahmoud
[EP-11] Edit Grades	Course details	Hala Mahmoud
[EP-12] Create Attendance	Course details	Yara Abd El Rahman
[EP-13] View Instructor Attendance	Course details	Yara Abd El Rahman
[EP-14] View Student Attendance	Course details	Yara Abd El Rahman
[EP-15] Create Announcement	Course details	Yara Abd El Rahman
[EP-16] View Announcement	Course details	Yara Abd El Rahman
[EP-17] Welcome page	Course details	Yousef Reda
[EP-18] Create Course	Course details	Yousef Reda
[EP-19] Course Information	Course details	Yousef Reda
[EP-20] Create Lecture	Course details	Yousef Reda
[EP-21] Course Lecture	Course details	Yousef Reda
[EP-22] Lecture Details	Course details	Abanoub Fayez
[EP-23] Create Quiz	Course details	Ahmed Fathy
[EP-24] Add Quiz Questions	Course details	Ahmed Fathy
[EP-25] Course Quizzes	Course details	Ahmed Fathy
[EP-26] Quiz Main Details	Course details	Ahmed Fathy
[EP-27] answer Questions	Course details	Ahmed Fathy
[EP-28] Quiz Result	Course details	Yara Abd El Rahman
[EP-29] Course Exams	Course details	Ahmed Fathy
[EP-30] Upload attachment	Attachment	Abanoub Fayez

TOOLS

INTELLIJ IDEA:

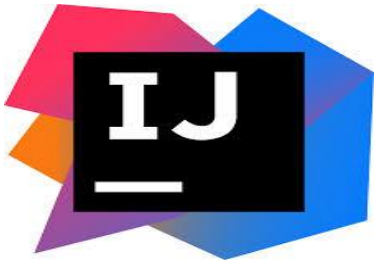


Figure 9 Intellij

It's an IDE (Integrated Development Environment) which we use to write our code and develop our system.

Features:

1. Supports many file types (Java, HTML, TS)
2. Easy to refactor, support.
3. Has plugin for Source Control
4. Has plugin for executes SQL queries on Database.

MYSQL WORKBENCH

Figure 10 my SQL workbench

MySQL Workbench is a client software for database designing and modeling for MySQL Server relational database. It facilitates creation of new physical data models and modification of existing MySQL databases with reverse/forward engineering and change management functions.

TORTOISESVN :

"TortoiseSVN is a client software used for the source control on Windows. It is based on Apache™ Subversion (SVN) ®"

Figure 11 Tortoize Svn

Features:

1. Provides a nice and easy user interface for Subversion.
2. Developed under the GPL. Which means it is completely free for anyone to use, including in a commercial environment, without any restriction.
3. The source code is also freely available, so you can even develop your own version if you wish to. Since it's not integration for a specific".

NODE JS

Figure 12 nodejs

It's a cross platform JavaScript runtime built on Chrome's V8 JavaScript engine, it executes JavaScript code outside of a browser.

Node.js let developers use JavaScript to write command line tools and run server side scripts to produce dynamic web page content before the page is rendered by the web browser.

Consequently, Node.js represents a "JavaScript everywhere" node.js includes npm which is considered a large repository with many libraries for Node based projects.

NPM includes two distinct components:

1. Website: used for searching for packages
2. Command Line Interface (CLI): used for run & build the project, It's commands executes in terminal:
   - npm install <library-name> → to install libraries.
   - npm start → to run the angular code application

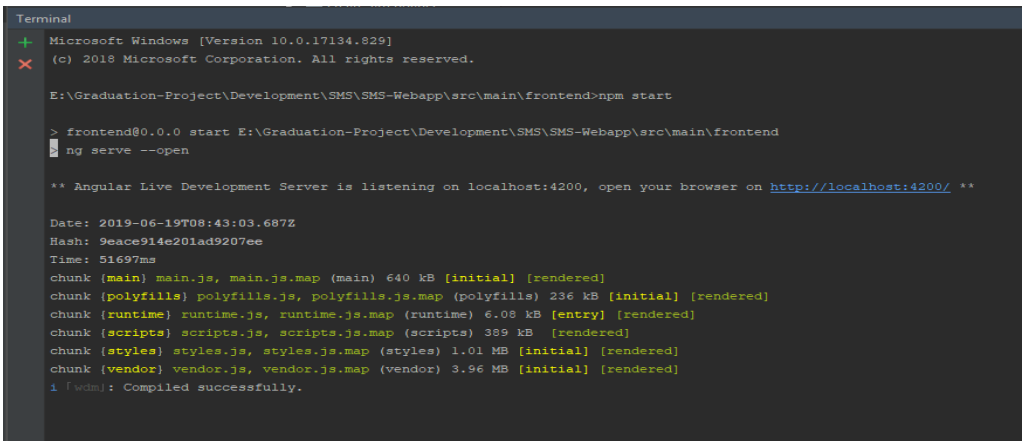


Figure 13CLI

THE JAVA DEVELOPMENT KIT (JDK)

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications. It contains JRE + development tools. JDK is used for implementing any one of the below Java Platforms released by Oracle Corporation:

- Standard Edition Java Platform
- Enterprise Edition Java Platform
- Micro Edition Java Platform

The JDK contains a Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. used in Java Application development.

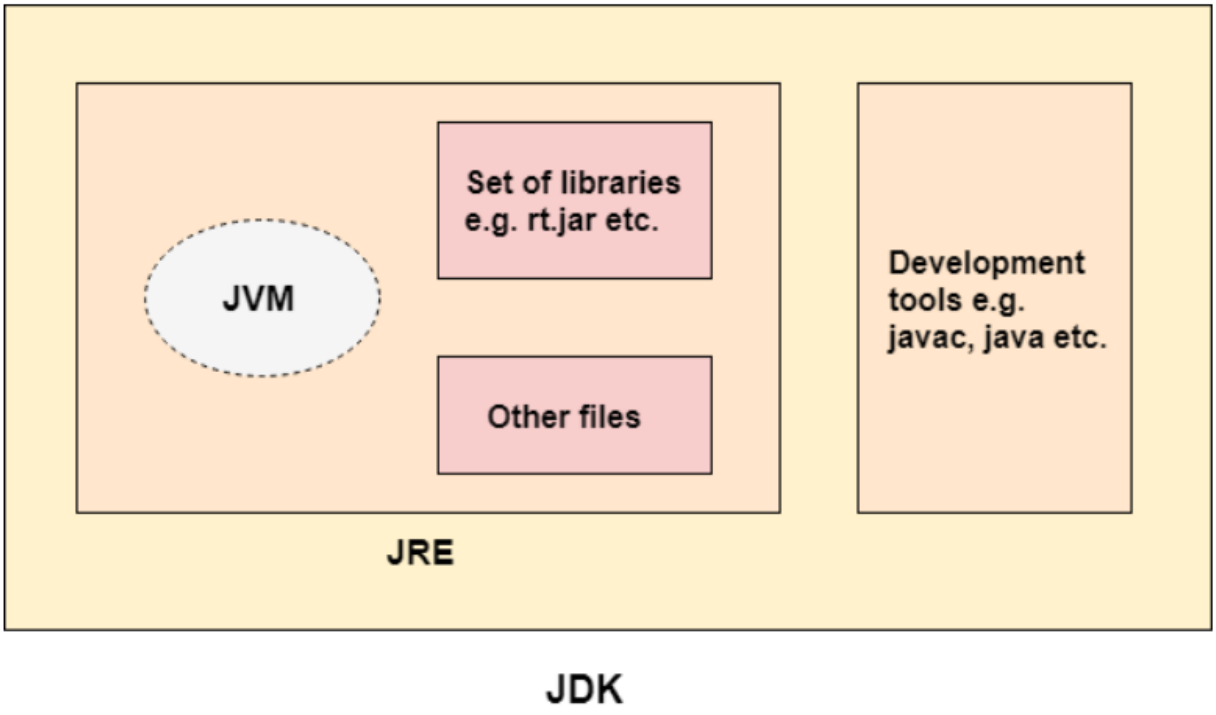


Figure 14 JDK

VERSION CONTROL:

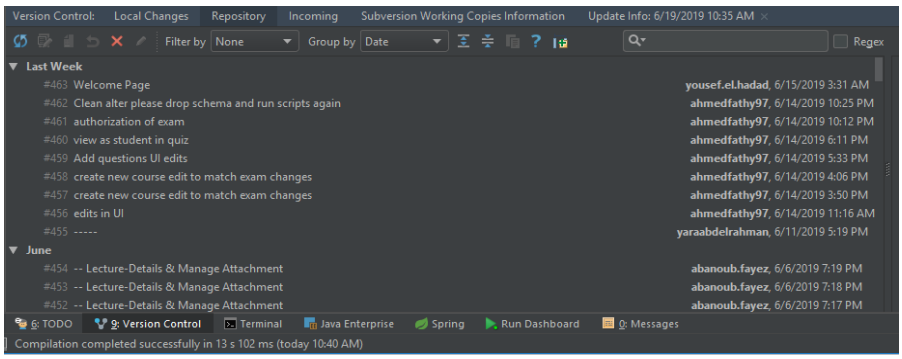


Figure 15 history

It is used to manage and control the changes that occur in the code and documents, which is also known as reversion control or source control, these changes are finally merged into several versions.

TECHNOLOGIES

- **Angular 6** (components , Structural Directives , Service , Routing , Guards , Interceptor)

  There are five major releases of Angular. The first version that was released is Angular 1, which is also called AngularJS. Angular 1 was followed by Angular 2, which came in with a lot of changes when compared to Angular 1.

  The structure of Angular is based on the components/services architecture.

Figure 16 angular

- **Common features in version 5 and 6**

  1- HTTPClient API.

  HTTPClient API was introduced to deprecate the HTTP library. It is much faster, secure and efficient than HTTP library.

  2- Multiple export aliases.

  3- Internationalized Pipes for Number, Date, and Currency.

  4- Build Optimizer

  Build Optimizer introduced. It optimizes the build size and improves the application speed. Angular CLI uses Build Optimizer automatically.

  5- Improved Compiler

  Compiler from Angular 5 leading for faster compilation. Compiler uses Typescript transforms.

- **The new features added to Angular**

  6- Updated Angular CLI, Command Line interface

  New commands added, like ng-update to migrate from previous version to current version. ng-add to quickly add application features to make application a progressive web apps.

  7- Updated CDK, Component Development Kit.

  Supports creating custom UI elements without need of angular material library. Supports responsive web design layouts. Supports overlay packages to create pop-ups.

8- Updated Angular Material

New Tree component added, mat-tree, a styled version and cdk-tree, a unstyled version, to represent a hierarchical structure like tree.

9- Angular Element

Allows Angular Components to be published as Web Components which can then be used in any HTML page. Using Angular Element package, native custom elements can be created easily.

COMPONENTS

Major part of the development with Angular 6 is done in the components. Components are basically classes that interact with the .html file of the component, which gets displayed on the browser. The file structure has the app component and it consists of the following files.

app.component.css

app.component.html

app.component.ts

STRUCTURAL DIRECTIVES

Structural directives are responsible for HTML layout. They shape or reshape the DOM's structure , by adding ,removing ,or manipulating elements. Structural directives have a * sign before the directive. For example, ***ngIf** and ***ng For**.

DOM: Document Object Model (as body , div , ….) is the way Javascript sees its containing pages data. It is an object that includes how the HTML/XHTML/XML is formatted, as well as the browser state.

ROUTING

Routes tell the router which view to display when a user clicks a link or pastes a URL into the browser address bar.

A typical Angular Route has two properties:

- path: a string that matches the URL in the browser address bar.
- component: the component that the router should create when navigating to this route.

TYPESCRIPT (TS)

TypeScript is JavaScript plus some additional features.

TypeScript is a primary language for Angular application development. It is a superset of JavaScript with design-time support for type safety and tooling.

Browsers can't execute Typescript directly. Typescript must be convert into JavaScript using the –**TSC**– compiler, which requires some configuration**.**

Figure 17 type script

SPRING BOOT

Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that we can "**just run"** with minimum configurations without the need for an entire spring configuration setup. Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files). Spring boot help us:

Figure 18 spring boot

- To avoid complex XML configuration in spring.
- To develop a production ready spring applications in an easier way.
- To reduce the development time and run the application independently.
- Offer an easier way of getting started with the application.
- To provides a powerful batch processing and manages REST endpoints.

SPRING JDBC

It is a powerful mechanism to connect to the database and execute SQL queries. It internally uses JDBC API .It provides us with methods to write the queries directly so it help us to save a lot of time.

Spring framework provides following approaches for JDBC database access:

- JdbcTemplate
- NamedParameterJdbcTemplate
- SimpleJdbcTemplate
- SimpleJdbcInsert and SimpleJdbcCall.

- **Basic Queries**
  - This is the central framework class that manages all the database communication and exception handling.
  - is the main API through which we'll access most of the functionality that we're interested in:
    1. Creation and closing of connections.
    2. Executing statements and stored procedure calls.
    3. Iterating over the Result Set and returning results.

RESTFUL WEB SERVICES (API)

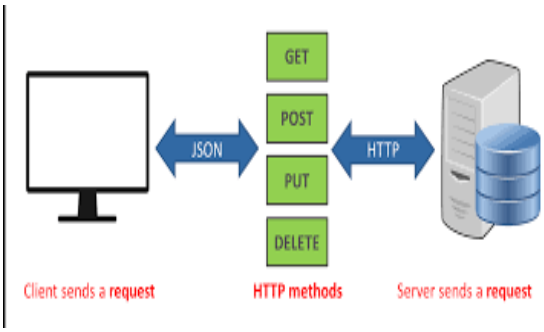


Figure 19 Rest API

- A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

- Web services based on REST Architecture are known as RESTful web services. These web services uses HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation such as JSON and set of HTTP Methods.

  a) HTTP methods.

  There are four HTTP methods are commonly used in REST based architecture:

  1. **GET**: Provides a read only access to a resource.
  2. **POST**: Used to create a new resource.
  3. **DELETE**: Used to remove a resource.
  4. **PUT**: Used to update existing resource or create a new resource.

b) Jersey -RESTful Web Services in Java

In order to simplify development of restful Web services and their clients in Java, a standard and portable JAX-RS API has been designed.

- **Jersey RESTful Web Services framework** is open source, production quality, framework for developing RESTful Web Services in Java that provides support for JAX-RS APIs and serves as a JAX-RS (JSR 311 & JSR 339) Reference Implementation. Jersey framework is more than the JAX-RS Reference Implementation and provides its own API that extend the JAX-RS toolkit with additional features and utilities to further simplify RESTful service and client development.
  - **Goals of Jersey summarized in the following points:**
    a. Track the JAX-RS API and provide regular releases of production quality Reference Implementations that ships with Glassfish;
    b. Provide APIs to extend Jersey & Build a community of users and developers; and finally
    c. Make it easy to build RESTful Web services utilizing Java and the Java Virtual Machine.

HTML5

HTML5 stands for Hyper Text Markup Language that describes the structure of a Web page and consists of a series of elements Those tell the browser how to display the content and are represented by tags. This tags considered as label pieces of content such as "heading" <h>, "paragraph" <p>, "table" <table>, and so on.

Figure 20 Html&CSS

CSS3

CSS stands for **C**ascading **S**tyle **S**heets that describes how HTML elements are to be displayed on screen, paper, or in other media**.**

PROJECT ARCHITECTURE

Project Architecture means what are the layers in our project with flow diagram .each tier can be referred as a 'layer'**. E-Peak** project contains three tiers which is consider three layers like angular layer , spring layer and data layer , then we draw the all layers flow.

E-Peak System consist of three tiers Angular, Spring and Database:

- Front-End Layer (Angular)
- Back-End Layer (API)
- Database Layer (MySQL)

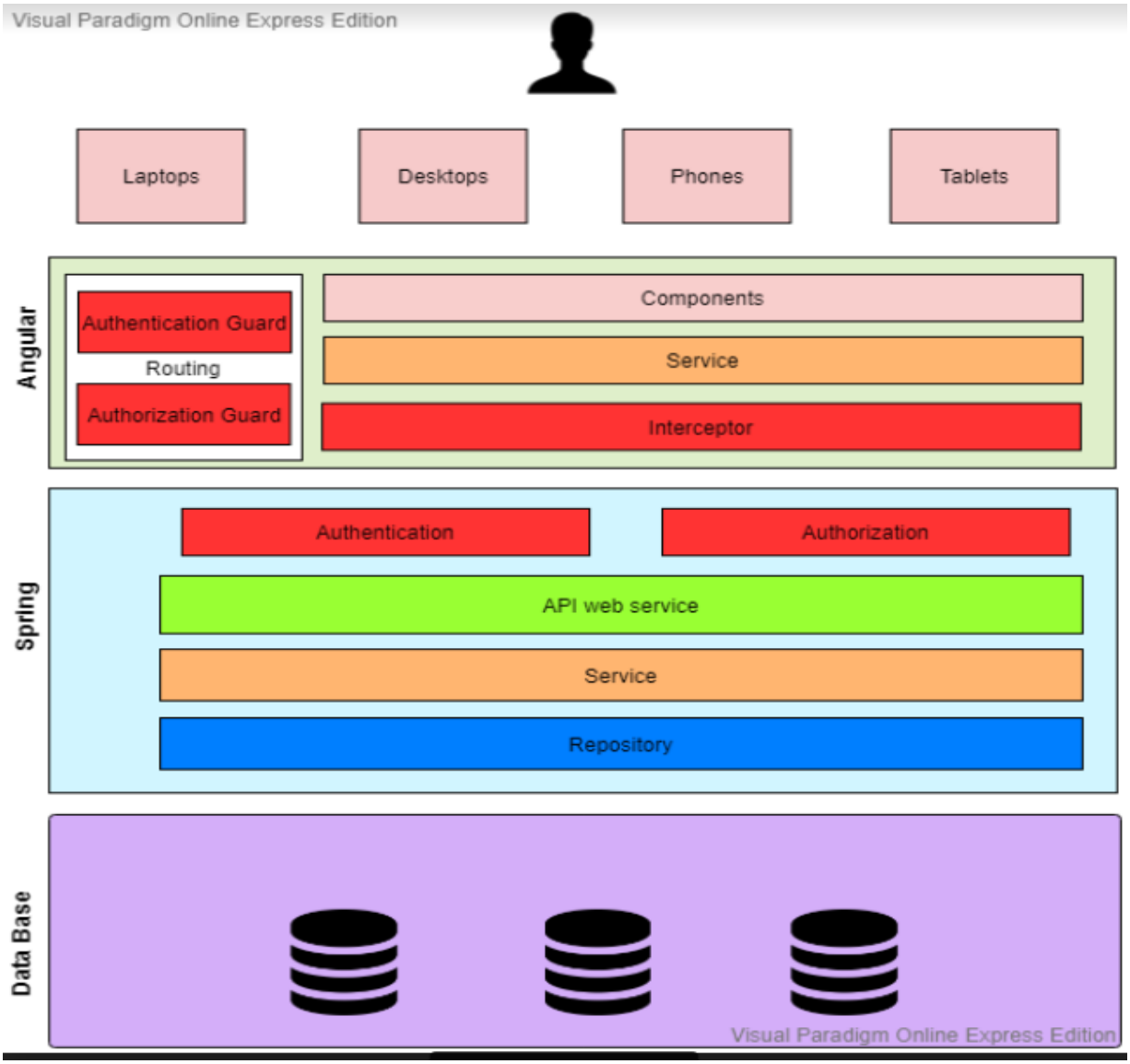


Figure 21 project architecture

FRONT-END LAYER

It is also known as Client Layer. Top most layer of an application. The main functionality of this layer is to communicate with Application Layer.

The Flow of this tier:

- User using UI Application to make actions with buttons or fill a form with data.
- The HTML form store data as an object which is carry it to service.
- At Service Class Using REST API web service data send to Spring Layer.

For Example:

Create Course Component when an end user could see form and buttons to enter course data and to click on submit.

AUTHENTICATION & AUTHORIZATIONS IN FRONT-END LAYER:

- **Interceptor :**

  Interceptors provide a mechanism to intercept and/or mutate outgoing requests or incoming responses. They are very similar to the concept of middleware with a framework like Express, except for the frontend. Interceptors can be really useful for features like caching and logging.

- **Guards :**

we used guard routing (as CanActivate) in authentication and authorization to control privilege of user such the following cinditions ;

- Maybe the user must login (authenticate) first.
- Perhaps the user has logged in but is not authorized to navigate to the target component.
- We might ask the user if it's OK to discard pending changes rather than save them.

There are four different types of Guards:

1. **CanActivate :** Checks to see if a user can visit a route.
2. **CanActivateChild:** Checks to see if a user can visit a routes children.

BACK-END LAYER

As create course component example, once user click on submit spring layer interact with data base layer and sends course data. It controls an application's functionality by performing detailed processing. This layer acts as mediator between the Presentation or UI and the Database layer.

The Flow of this tier:

1. Now data are located in service at Controller or Resource class which able to send it to another Service class using object of Model class carried same front-end data.
2. The Service class make a logic operations on data before sending it to Repository.
3. Then at Repository Class there are service take which his parameter your data and its body is SQL Query which do the excellent action user had chosen.

AUTHENTICATION & AUTHORIZATION IN BACK-END LAYER:

- **Authentication filter :** In token-based authentication, the client exchanges *hard* credentials (such as username and password) for a piece of data called *token*. For each request, instead of sending the hard credentials, the client will send the token to the server to perform authentication and then authorization.

Figure 22 authentication

In a few words, an authentication scheme based on tokens follow these steps:

1. The client sends their credentials (username and password) to the server.
2. The server authenticates the credentials and, if they are valid, generate a token for the user.
3. The server stores the previously generated token in some storage along with the user identifier and an expiration date.
4. The server sends the generated token to the client.
5. The client sends the token to the server in each request.
6. The server, in each request, extracts the token from the incoming request. With the token, the server looks up the user details to perform authentication.
    - If the token is valid, the server accepts the request.
    - If the token is invalid, the server refuses the request.
7. Once the authentication has been performed, the server performs authorization.
8. The server can provide an endpoint to refresh tokens.

- **Authorization filter:**
  This filter is used to determine if the current user has the privilege to access the requested view or make the requested action. If the user has privilege to access This request ,the server response with ok. Else ,the server response with error 401 unauthorized.

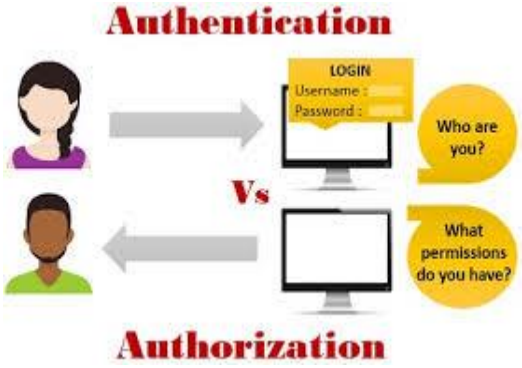


Figure 23 authorization

DATATIER LAYER

The Data is stored in this layer. Spring layer communicates with Database layer to retrieve or store data.

It contains a methods that connects the database and performs required action.

E.g.: INSERT,UPDATE, DELETE and SELECT.

SUMMARY OF ARCHITECTURE:

E-Peak Architecture diagram show system flow from action which is human user take to the result and changes in database. And in our system we have three roles e.g.: Instructor, Student and System Admin. So we should ensure that which role of users is signed in to limit his actions.

The Front End layer which is Web Application build in Angular interaction with authentication and authorization services to define the user.

Then, any action user takes with UI translated to second phase Back-End layer by using web service like API. In controller the business logic operations are performed and send to Repository which interaction with Database tier.

In Database tier, there are some queries are carried from Repository or Back-End Layer. These queries are executed and change in current data.

IMPLEMENTAION DETAILS

COURSE RES

Story	**[EP-1] Create Course**	**Controller**	CourseRes	**Function**	createNewCourse()
Method	**POST**	**URL**	**/api/courses/**		
Description	Instructor fill the information of the Course in the System and submit the form to add new Course.				
Flow	1. Validate the Input of the form 2. Check the instructor ID 3. Insert the Data of the Course				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**Action**	**ADD-COR**
Instructor	**View**	**ADD-COR**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
CousreDTO	PathParameter	details	It contains a course information from HTML form.

Story	**[EP-19] Course Information**	**Controller**	CourseRes	**Function**	viewCourse()
Method	**GET**	**URL**	**/api/courses/{corID}**		
Description	When user click on course info the system send course id and get course data from Data Base.				
Flow	1. Send course ID 2. Select the course data 3. Return data to HTML and present it .				

Authorization		
Role	**Privilege Type**	**Privilege value**
Any User	**Action**	**-**
Any User	**View**	**-**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
CorID	PathParmter	Int	It contains a course id which used to select course data.

Story	**[EP-20] Create Lecture**	**Controller**	CourseRes	**Function**	createNewLecture()
Method	**POST**	**URL**	**/api/courses/{courseID}/lecture**		
Description	Instructor fill the information of the lecture in the System and submit the form to add new lecture				
Flow	1. Validate the Input of the form 2. Check the instructor ID 3. Insert the Data of the Lecture				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**Action**	**ADD-LEC**
Instructor	**View**	**ADD-LEC**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
LectureDTO	PathParmter	LectureDTO	Contain a Lecture information from HTML form.

Story	**[EP-6] All Courses**	**Controller**	CourseRes	**Function**	findAllCourses()
Method	**Get**	**URL**	**/api/course/all**		
Description	This function gets all courses from database and display it for student to enable it to make join in any course				
Flow	1-call service(**findAllCourse**) in (**Course Res**) in **findAllCourses** function that's method is **GET** which it's dataType is (**CourseResultSet**) use this model to return a list of course information(CourseVto) and used in pagination 2-usig services to call my repositry This services (**in CourseSer**)is **findAllCourse** which it's dataType is (**CourseResultSet**) use this model to return a list of course information(CourseVto) and used in pagination 3-(**in CourseRep**)I used a select query to get all courses with specific information such as (id,duration,course-name,start date,end date, description, instructor name) from database				

Authorization		
Role	**Privilege Type**	**Privilege value**
instructor	**view**	**COURSE_LIST**
student	**view**	**COURSE_LIST**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
pageNum	Query parameter	int	It contains the num of page that these courses display in it.

Story	**[EP-5] My Courses**	**Controller**	CourseRes	**Function**	myCourse
Method	**GET**	**URL**	**/api/course/**		
Description	This function check first who is login if it was instructor display courses that's who create If it was student display courses that he enrolls in it and each student can see its courses				
Flow	1-call service(**myCourse**) in (**Course Res**) in **myCourses** function that's method is **GET** which it's dataType is (**CourseResultSet**) use this model to return a list of course information(CourseVto) and used in pagination 2-usig services to call my repositry -This services (**in CourseSer**)is **myCourse** which it's dataType is (**CourseResultSet**) use this model to return a list of course information(**CourseVto**) and used in pagination -there is in it check to know who is log in instructor or student to determine which repositry will call it . 3-(**in CourseRep**) I used a select query to get all instructor courses (**findAllInstructorCourse**)and select query to get all student courses(**findAllStudentCourse**) with specific information such as (id,duration,course-name,start date,end date, description, instructor name) from database.				

Authorization		
Role	**Privilege Type**	**Privilege value**
instructor	**view**	**MY_COURSE**
student	**view**	**MY_COURSE**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
pageNum	Query parameter	int	It contains the num of page that these courses display in it.
currentUser	Path parameter	int	It contains value we know from it who is login (student or instrctor)

Story	**[EP-7] view instructor grade**	**Controller**	CourseRes	**Function**	getCourseGrade
Method	**GET**	**URL**	**/api/course/{courseID}/grade**		
Description	This function display the grade of all student to the instructor				
Flow	• Query return list of students with grades (first name , last name , mid 1 , final) • Then display them to instructor				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**View**	**ADD_GRADE**
Student	**View**	**ADD_GRADE**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**

corID	Path parameter	int	It contain a number to tell us which is couse .
pageNum	Queryparameter	int	It contains the num of page that these courses display in it.

Story	**[EP-8] View Student grade**	**Controller**	CourseRes	**Function**	getCourseGrade
Method	**GET**	**URL**	**/api/course/{courseID}/grade**		
Description	This function display grade for student by its id .				
Flow	• Query return specific student with grades (first name , last name , mid 1 , final) • Then display them to specific .				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**View**	**ADD_GRADE**
Student	**View**	**ADD_GRADE**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
corID	Path parameter	int	It contain a number to tell us which course is choosen .
pageNum	Queryparameter	int	It contains the num of page that these grade display in it.

Story	**[EP-7] view instructor quiz grade**	**Controller**	CourseRes	**Function**	getQuizGrade
Method	**Get**	**URL**	**/api/course/{courseID}/grade/quizes**		
Description	This function display the grade quizes of all student to the instructor				
Flow	1. Query return list of students with Quiz grades (first name , last name , StdScore , totalScore) 2. Then display them to instructor				

Authorization		
Role	**Privilege Type**	**Privilege value**
instructor	**View**	**ADD_GRADE**
Student	**View**	**ADD_GRADE**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
corID	Path parameter	int	It contain a number to tell us which course is choosen .
pageNum	Query parameter	int	It contains the num of page that these grade display in it.

Story	**[EP-7] view student quiz grade**	**Controller**	CourseRes	**Function**	getQuizGrade
Method	**Get**	**URL**	**/api/course/{courseID}/grade/quizes**		
Description	This function display the grade quizes of to each student				
Flow	1. Query return specific students with Quiz grades (first name , last name , StdScore , totalScore) 2. Then display it to student				

Authorization		
Role	**Privilege Type**	**Privilege value**
instructor	**View**	**ADD_GRADE**
student	**View**	**ADD_GRADE**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
corID	Path parameter	int	It contain a number to tell us which course is choosen
pageNum	Query parameter	int	It contains the num of page that these grade display in it.

Story	**[EP-11] View instructor attendance**	**Controller**	CourseRes	**Function**	getCourseAttendance ()
Method	**GET**	**URL**	**/api/course/courseID/attendance**		
Description	Display attendance sheet of students for instructor				
Flow	• We have 2 Queries 1- Query return list of student (**findAllCourseStudents(id)**) . 2- Query return list of student attendance (first name , last name , date , is_attend) 3- Transfer this lists into pivot table				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**View**	**ADD_ATTENDANCE**

REST Parameter				
Attribute	**Parameter Type**	**Data Type**	**Description**	
courseID	Payload	INT	It contains id of course	

Story	**[EP-11] View student attendance**	**Controller**	CourseRes	**Function**	getCourseAttendance( )
Method	**GET**	**URL**	**/api/course/courseID/attendance**		
Description	Display attendance of specific student				
Flow	• We have 2 Queries 1- Query return one student depend on its id. 2- Query return list of student attendance (first name , last name , date , is_attend) 3- Transfer this list into pivot table				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**View**	**ADD_ATTENDANCE**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
courseID	Payload	INT	It contains id of course

Story	**[EP-13] Create Announcement**	**Controller**	CourseRes	**Function**	createAnnouncement()
Method	**POST**	**URL**	**/api/course/courseID/newAnnouncment**		
Description	Instructor fill the information of Announcement he want to tell students about it				
Flow	1. It is checked if instructor has been logged in the course is the same instructor who created it (**by ID**) 2. Insert the Data of the Announcement in database (insert query in repository that add title and description about announcement in database)				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**Action**	**COR_ADD_ANNOUNCEMENT**
Instructor	**View**	**COR_ADD_ANNOUNCEMENT**
Student	**View**	**COR_ADD_ANNOUNCEMENT**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
request	Payload	ContainerRequestContext	It contains the current user
courseID	PathParameter	INT	It contains a course information from HTML form.
announcement	Payload	Announcement	It contains title and description of announcement

Story	**[EP-14] View Announcement**	**Controller**	CourseRes	**Function**	getCourseAnnouncments()
Method	**GET**	**URL**	**/api/course/courseID/announcmentList**		
Description	Display Announcements to students				
Flow	Query return announcement that instructor want student to see it with date				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**View**	**COR_ADD_ANNOUNCEMENT**
Student	**View**	**COR_ADD_ANNOUNCEMENT**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
cousreID	PathParameter	INT	It contains id of course.
pageNum	QueryParameter	INT	It contains page number.

Story	**[EP-18] create quiz**	**Controller**	**CourseRes**	**Function**	createQuiz(int courseID)
Method	**post**	**URL**	Api/Course/{courseID}/newQuiz		
Description	This API is used by instructor to add quiz to his course.				
Flow	1- API receive quiz data from user. 2- Apply some validations in service. 3- Insert quiz data into database.				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**View**	**ADD-QUIZ**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
quizData	---	QuizDTO	It is object from QuizDto model to collect quiz data.
courseID	Path parameter	Int	It carry the value of course number.

Story	**[EP-1] Course's Lectures**	**Controller**	CourseRes	**Function**	getCourseLectures()
Method	**GET**	**URL**	**/api/courses/{corID}/lectures**		
Description	When user click on lectures button the system send course id and get list of lectures from Data Base.				
Flow	1. Send course ID 2. Select the course's lectures list from data base. 3. Return list to HTML and present it .				

Authorization		
Role	**Privilege Type**	**Privilege value**
Any User	**Action**	**-**
Any User	**View**	**-**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
corID	PathParmter	Int	It contains a course id which used to select course's lectures.
pageNum	Query parameter	int	It contains the num of page that these courses display in it.

Story	**[EP-20] Course Quizes**	**Controller**	**CourseRes**	**Function**	getCourseQuizes()
Method	**Get**	**URL**	API/Course/{courseID}/Quizes		
Description	This API is used by student and Instructor to view list of course quizes .				
Flow	1. Receive courseID from client application. 2. Apply some validations on service. 3. Retrieve list of quizes from database.				

Authorization		
Role	**Privilege Type**	**Privilege value**
student	**view**	**Course_Quizes**
instructor	**view**	**Course_Quizes**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
corID	Path parameter	int	It carry the value of course number.

Story	[EP-22] Course exams	Controller	CourseRes	Function	getCourseExams()
Method	Get	URL	API/Course/{courseID}/Exams		
Description	This API is used by student and Instructor to view list of course quizes .				
Flow	1. Receive courseID from client application. 2. Apply some validations on service. 3. Retrieve list of exams from database.				

Authorization		
Role	Privilege Type	Privilege value
student	view	Course_EXAMS
instructor	view	Course_ EXAMS

REST Parameter			
Attribute	Parameter Type	Data Type	Description
corID	Path parameter	int	It carry the value of course number.

QUIZ RES

Story	[EP-19] add questions	Controller	QuizRes	Function	createQuizQuestions()
Method	post	URL	Api/Quiz/{quizID}/questions		
Description	This API is used by instructor to add questions to quiz.				
Flow	1 .receive list of questions from angular applications. 2. apply some validations om service . 3.insert quiz questions into database .				

Authorization		
Role	Privilege Type	Privilege value
Instructor	View	ADD-QUESTIONS

REST Parameter			
Attribute	Parameter Type	Data Type	Description
quizID	Path parameter	Int	It carry the value of quiz number.
questions	---	LIST<QuestionsDto>	It is object from QuestionDto model to collect questions .

Story	**Quiz state**	**Controller**	**QuizRes**	**Function**	getQuizState()
Method	**Get**	**URL**	API/quiz/{quizID}/state		
Description	This API is used to get the state of the quiz to determine which page will appear to user that is logged in now.				
Flow	1. Receive current user that is active now. 2. According to user and its role the method return number if state				

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
currentUser	Context	UserVto	It carry information about the current user that is active now
quizID	Path parameter	Int	It carry the value of quiz number.

Story	**Quiz Information**	**Controller**	**QuizRes**	**Function**	getQuizInfo()
Method	**Get**	**URL**	API/quiz/{quizID}/quizDetails		
Description	This API is used by instructor to view details about the quiz such that the number of students that answered the quiz				
Flow	1. Receive quizID from web application. 2. Apply some validations on service. 3. Retrieve information of the quiz from database				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**View**	**QUIZ_INFO**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
quizInfo	---	QuizInfoVTO	It carry object of quiz information.
quizID	Path parameter	Int	It carry the value of quiz number.

Story	**Quiz questions**	**Controller**	**QuizRes**	**Function**	getQuizQuestiosn()
Method	**Get**	**URL**	API/quiz/{quizID}/questionsView		
Description	This API is used by student to view quiz questions available for answering.				
Flow	1. Receive quizID from client application. 2. Apply some validations on service. 3. Retrieve information of the quiz from database.				

Authorization		
Role	**Privilege Type**	**Privilege value**
Student	**View**	**QUIZ_INFORMATION**
instructor	**View**	**QUIZ_INFORMATION**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
quizID	Path parameter	Int	It carry the value of quiz number.
quizQuestions	----	List<QuestionsVto>	Carry List of questions for specific quiz

Story	**submitQuizAnswers**	**Controller**	**quizRes**	**Function**	submitAnswer()
Method	**Post**	**URL**	API/quiz/{quizID}/Answer		
Description	This API is used by student to submit quiz questions after answer.				
Flow	1. Receive quizID from client application. 2. Receive current user also 3. Apply some validations on service. 4. Insert list of answers to database				

Authorization		
Role	**Privilege Type**	**Privilege value**
Student	**View**	**Submit_Answer**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
quizID	Path parameter	Int	It carry the value of quiz number.
studentAnswere	-----	List<StudentAnswer>	Carry List of student answere for specific quiz

Story	**Close quiz**	**Controller**	**QuizRes**	**Function**	closeQuiz()
Method	**Update**	**URL**	API/quiz/{quizID}/close		
Description	This API is used by instructor to close quiz after deadline.				
Flow	1. Receive quizID from client application 2. Apply some validations on service. 3. Update is_closed column to true .				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**action**	**CLOSE_QUIZ**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
quizID	Path parameter	Int	It carry the value of quiz number.

Story	**Quiz Result**	**Controller**	**QuizRes**	**Function**	getQuizResult()
Method	**Get**	**URL**	API/quiz/{quizID}/result		
Description	This API is used by student to view details about his answer of quiz.				
Flow	1. Receive quizID from client application 2. Apply some validations on service. 3. Retrieve result form database.				

Authorization		
Role	**Privilege Type**	**Privilege value**
student	**View**	**QUIZRESULT**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
quizID	Path parameter	Int	It carry the value of quiz number.

ATTANDANCE RES

Story	**[EP-11] Create Attendance**	**Controller**	Attendance Res	**Function**	createAttendance Sheet ()
Method	**POST**	**URL**	**/api/attendance/courseID/new**		
Description	Instructor fill the information of the Lecture on specific course (attendance date , attendance) to add new attendance sheet on course				
Implementation	1. It is checked if instructor has been logged in the course is the same instructor who created it (**by ID**) 2. It is checked if the date has been recorded in the absence sheet before or not • If has been recorded , return list of students who attended in this date to allow the instructor to modify student's attendance • If not , return list of students enrolled in this course 3. Determines the status of the student (attend – absent)				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**Action**	**ADD_ATTENDANCE**
Instructor	**View**	**ADD_ATTENDANCE**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
Request	Payload	`ContainerRequestContex`	It contains the current user
courseID	PathParameter	INT	It contains course id
attendanceDate	Payload	AttendanceDTO	It contains date of lecture

GRADE RES

Story	**[EP-9] Edit grade**	**Controller**	GradeRes	**Function**	EditGradeSheet
Method	**Post**	**URL**	**/api/grade/{courseID}/new**		
Description	This function enable instructor to make edit on the grade of the exams				
Flow	1-after an instructor make edit on the grade 2-the query make a post to this value in database.				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**action**	**COR_ADD_GRADE**
Instructor	**View**	**ADD_GRADE**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
corID	Path parameter	int	It contain a number to tell us which is course to get the grade of it to make edit on it.
GradeType	Query parameter	string	Specify which type of grade is select to make edit on it.

USER RES

Story	[EP-3]User Profile	Controller	UserRes	Function	`findUserByID()`
Method	**GET**	**URL**	**api/User/{userID}/profile**		
Description	View profile data of the current logged in user.				
Flow	1. This services receive user id. 2. If user is found, select it's data from database. 3. Return user data in userVTO object.				

Authorization		
Role	**Privilege Type**	**Privilege value**
Any Role	**Views**	**USER_PROFILE**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
userID	PathParam	Int	----

Story	[EP-4]Edit user	Controller	UserRes	Function	`updateUserVto()`
Method	**POST**	**URL**	**api/User/{userID}/edit**		
Description	User fill the information about himself in the edit form and click save to save this information in database				
Flow	1. Validate the Inputs of the form. 2. If inputs are ok, save data in database. 3. Return user data in userVTO object.				

Authorization		
Role	**Privilege Type**	**Privilege value**
Any Role	**Views**	**USER_EDIT**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
userID	PathParam	Int	

Story	[EP-]User List	Controller	UserRes	Function	`findAllUsers()`
Method	**GET**	**URL**	**api/User/find**		
Description	View all users in the system.				
Flow	1. When admin click on any link in full name. 2. Admin will go to the user's profile and will know all information about this user. 3. Return user data in userVTOList object.				

Authorization		
Role	**Privilege Type**	**Privilege value**
Any Role	**Views**	**USER_LIST**

LECTURE RES

Story	**[EP-25] Lecture Details**	**Controller**	LectureRes	**Function**	listLectures()
Method	**GET**	**URL**	**/api/lecture/{lecID}**		
Description	Instructor or student can view lecture details				
Flow	4. Click on lecture label name 5. Retrieve lecture data 6. Redirects lecture details page				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**View**	**Lecture-details**
Student	**View**	**Lecture-details**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
	PathParam	LectureDTO	1. Receive LectureID 2. Retrieve list of lectures from database 3. Return the list

ATTACHMENT RES

Story	**[EP-26] Upload Files**	**Controller**	AttachmentRes	**Function**	uploadFile()
Method	**POST**	**URL**	**/api/attachment/file**		
Description	Instructor browse files on his computer and clicks on upload to save it on server				
Flow	7. Browse files 8. upload 9. receive response				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**Action**	**Add-materials**
Instructor	**View**	**Add-materials**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
File	QueryParam	Response	4. Receive file information and content 5. Save it in storage 6. Insert information the database 7. Return response with the state of file

Story	**[EP-27] Retrieve List of Files**	**Controller**	AttachmentRes	**Function**	listFiles()
Method	**GET**	**URL**	**/api/attachment/files**		

Description	Shows the list of files in the course or a specific lecture
Flow	1. Obtain Lecture ID 2. Get list of files of a specific lecture or course 3. Retrieve files for download

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**View**	**Lecture-details**
Student	**View**	**Lecture-details**

Story	**[EP-28] Download File**	**Controller**	AttachmentRes	**Function**	downloadFileById()
Method	**GET**	**URL**	**/api/attachment/downloadFile/{fileID}**		
Description	Instructor or student can download files				
Flow	1. click on file name label 2. browse to save content				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**Action**	**Download-materials**
Student	**Action**	**Download-materials**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
Resource ByteArrayResource	PathParam	Response	1. receive file id 2. retrieve file path of the file 3. return file as a resourceByteArray

Story	**[EP-29] remove File**	**Controller**	AttachmentRes	**Function**	removeFile()
Method	**DELETE**	**URL**	**/api/attachment/{fileID}**		
Description	Instructor can remove files from server				
Flow	1. list of files in the course or lecture 2. click on remove button				

Authorization		
Role	**Privilege Type**	**Privilege value**
Instructor	**Action**	**Add-materials**
Instructor	**View**	**Add-materials**

REST Parameter			
Attribute	**Parameter Type**	**Data Type**	**Description**
---	PathParam	Void	1. obtain file ID 2. delete file from storage 3. delete from database

DEPLOY PROJECT USING MAVEN

- **What is Maven?**

Maven is a large repository which have many libraries available for anyone use, Instead of downloading these libraries in the project, It provides us the ability to add these libraries as some dependencies in configuration file for the project (pom.xml), and It will download the libraries and import them in the project automatically. Maven also act as a building tool for the project and package it into deployable JAR Files.

- **What is POM?**

It's Project Object Model which is the fundamental unit of work in Maven, It is an XML file that contains dependencies of the project and configuration used by Maven to build the project.

When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal.

- **Maven Strcuture in the Project:**

  Root Project (Epeak).

  Backend (Epeak-Services).

  Frontend (Epeak-Webapp)

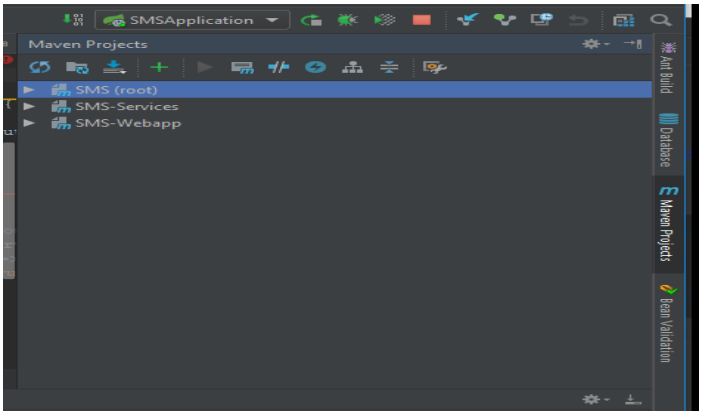


Figure 24 maven project

- **Root Project Configuration:**

  1. Declaring some variables such as JAR version & Build Path (build.path.directory)

  2. Declaring the Children Projects

  3. Copy Database Scripts to Build Path (Clean-Structure.sql) & (Clean-Data.sql) & (Alter.sql)

- **Backend Configuration:**

  Plugin for Building Backend Code in SMS-Services.jar

- **Frontend Configuration:**
  - Use Plugin to execute NPM commands to build angular code .
  - Plugin for Building SMS-Webapp.jar

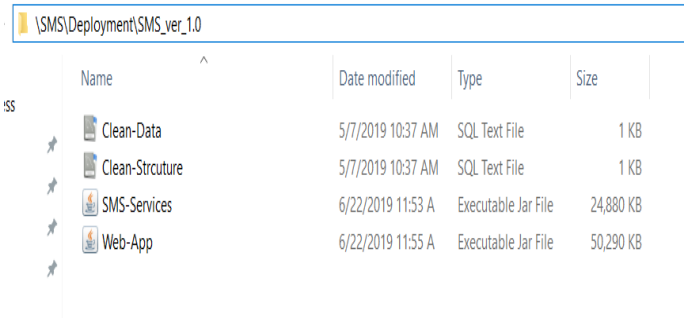


Figure 25 project after deployment

REFERENCES

https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/

https://www.javatpoint.com/spring-JdbcTemplate-tutorial

https://www.baeldung.com/spring-jdbc-jdbctemplate

https://www.tutorialspoint.com/restful/restful_introduction.html

https://jersey.github.io/

https://tortoisesvn.net/about.html

https://en.wikipedia.org/wiki/Java_Development_Kit

https://www.tutorialspoint.com/angular6/angular6_overview.html

https://codecraft.tv/courses/angular/routing/router-guards/

https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm

https://www.projectsmart.co.uk/project-management-time-estimates-and-planning.php.

https://en.wikipedia.org/wiki/Node.js.

https://www.guru99.com/introduction-to-mysql-workbench.html.

https://stackoverflow.com/questions/26777083/best-practice-for-rest-token-based-authentication-with-jax-rs-and-jersey