Specification of Requirements according to the IEEE 830 Standard

IEEE Std. 830-1998

October 22, 2008

Índex

1. Introduction

This project will discuss the development of an online certification platform for courses aimed at training employees, with the goal of using it in companies to train employees in businesses located in Tijuana

1.1. Purpose

To improve the skills and knowledge of personnel through the creation of courses, allowing employees to obtain certifications in their respective areas of work.

1.2. Scope of the System

- We will develop an online platform for the management, completion, and certification of courses aimed at employee training.

- The system will enable both company administrators and employees to interact, facilitating access to continuous learning, monitoring progress, and validating it with a certification

1.3. Definitions, Acronyms, and Abbreviations

FR: Function requirements

HIR: Hardware interface requirements

NFR: Non-functional requirements

1.4. References

This subsection will provide a complete list of all documents referenced in the SRS (Software Requirements Specification).

1.5. Document Overview

This document discusses the creation of a website for issuing company certifications

2. General Description

This is a system that allows companies to provide a training program for their staff. The expected user characteristics include anyone within a factory or company who needs training to improve in their work area. The restrictions are: the user must belong to a company or factory, and they must register to access the assigned courses. The assumed factors are that the users belong to a company or factory.

2.1. Product Perspective

This training page can be related to the company you are going to work with. For example, if you want to take a training course for the CFE, the courses will include videos and instructions on how to handle high voltage cables, safety measures, etc.

2.2. Product Features

The software will be used for employee training, tracking the progress of the employees, and the total points they obtained in each test, in order to obtain a certificate upon completing the training that validates the knowledge acquired.

Functional	Non-Functional
1. User Registration 2. Login 3. Course management 4. Progress Visualization 5. Exams or tests 6. Certificate Generation 7. Admin Panel	1. Security 2. Performance 3. Usability 4. Design 5. Order 6. Scalability 7. Maintainability

2.3. User Characteristics

They are all those who are in an institution or company. They are all those who are in a company or factory, who need training to improve their performance..

2.4. Restrictions

- You must be registered before taking a course or trying to log in.
- Only one registration per user

2.5. Assumptions and Dependencies

The website was made for any operating system that has a web browser. If you try to access it without a web browser it will not work, if the programming language is changed or modified it could present errors.

2.6. Future Requirements

Improve interfaces, make interfaces more intuitive

3. Requisitos Específicos

3.1 Requisitos comunes de las interfaces

qFunctional Requirements (FR)

Number of requirement	**FR1**
Requirement name	**User Registration**
Type	**Requirement**
Source	**Client/Users**
Priority	**High/Essential**

Description: The application must allow users to register by providing their first name, last name, email address, and a password. This information will be used to create a personal profile that stores their course progress.

Number of requirement	**FR2**
Requirement name	**Course Assignment**
Type	**Requirement**
Source	**Client**
Priority	**High/Essential**

Description: The application must allow administrators or trainers to assign courses to registered users. The assigned courses must be visible in the user's profile.

Number of requirement	**FR3**
Requirement name	**Task and Exam Management**
Type	**Requirement**
Source	**Users/Trainers**
Priority	**High/Essential**

Description: The application must allow users to view and complete tasks and exams associated with assigned courses.

Number of requirement	**FR4**
Requirement name	**Progress Tracking**
Type	**Requirement**
Source	**Users**
Priority	**Medium/Desired**

Description: The application must display the user's progress for each course, indicating the percentage completed, pending tasks, and completed exams.

Number of requirement	**FR5**
Requirement name	**Certificate Generation**
Type	**Requirement**
Source	**Client**
Priority	**High/Essential**

Description: The application must generate a digital certificate when a user

completes all assigned courses and passes the exams.

Non-Functional Requirements (NFR)

Number of requirement	**NFR1**
Requirement name	**System Performance**
Type	**Restriction**
Source	**Client**
Priority	**Medium/Desired**

Description: The system must run at good speed, with its important tables of the database indexed.

Number of requirement	**NFR2**
Requirement name	**Data Security**
Type	**Restriction**
Source	**Internal Standards**
Priority	**High/Essential**

Description: User personal data, including passwords and exam results, must be securely stored in their respective tables.

Number of requirement	**NFR 3**

Requirement name	**Scalability**
Type	**Restriction**
Source	**Client**
Priority	**Medium/Desired**

Description: The system's infrastructure must be scalable, allowing for an increase in users without significantly affecting performance.

Hardware Interface requirements (HIR)

Number of requirement	**HIR 1**
Requirement name	**User Device Compatibility**
Type	**Restriction**
Source	**Client**
Priority	**Medium/Desired**

Description: The application must be compatible with user devices such as desktops, laptops, tablets, and smartphones, supporting operating systems like Windows, macOS, iOS, and Android.

Number of requirement	**HIR2**
Requirement name	**Internet Connectivity**
Type	**Restriction**
Source	**Client**

Priority	**Medium/Desired**

Description: The application must function with a stable internet connection to allow users to download learning materials and stream video content without interruptions.

Software Interface Requirements (SIR)

Number of requirement	**SIR1**
Requirement name	**Database Management System (DBMS)**
Type	**Requirement**
Source	**IT Department**
Priority	**High/Essential**

Description: The system must interface with a MySQL database to store user information, course data, progress, and exam results.

Number of requirement	**SIR2**
Requirement name	**Web Browser Compatibility**
Type	**Requirement**
Source	**Client/Users**
Priority	**High/Essential**

Description: The application must be compatible with the latest versions of major web browsers, including Chrome, Firefox, Safari, and Edge, to ensure full functionality across devices.

3.1. Performance Requirements

- **Network connection speed:** The system must support Ethernet and wireless connections with a minimum speed of 10 Mbps to ensure smooth loading of multimedia content and exams.
- **Processing speed:** The server should process database requests..
- **System scalability:** The system must be scalable to accommodate more users and courses without affecting performance.

3.2. Design Restrictions

Web compatibility: The application must work across all modern web browsers (Chrome, Firefox, Safari, Edge) and mobile devices (iOS and Android).

User-friendly UI: The design must prioritize ease of use, ensuring intuitive navigation and minimizing the learning curve for non-technical users.

3.3. Other requirements

Maintainability: The system should be designed for easy maintenance, with clear documentation for developers to update or fix bugs efficiently.

Modularity: The system must be modular to allow future expansion, enabling easy addition of new features without significant rework.

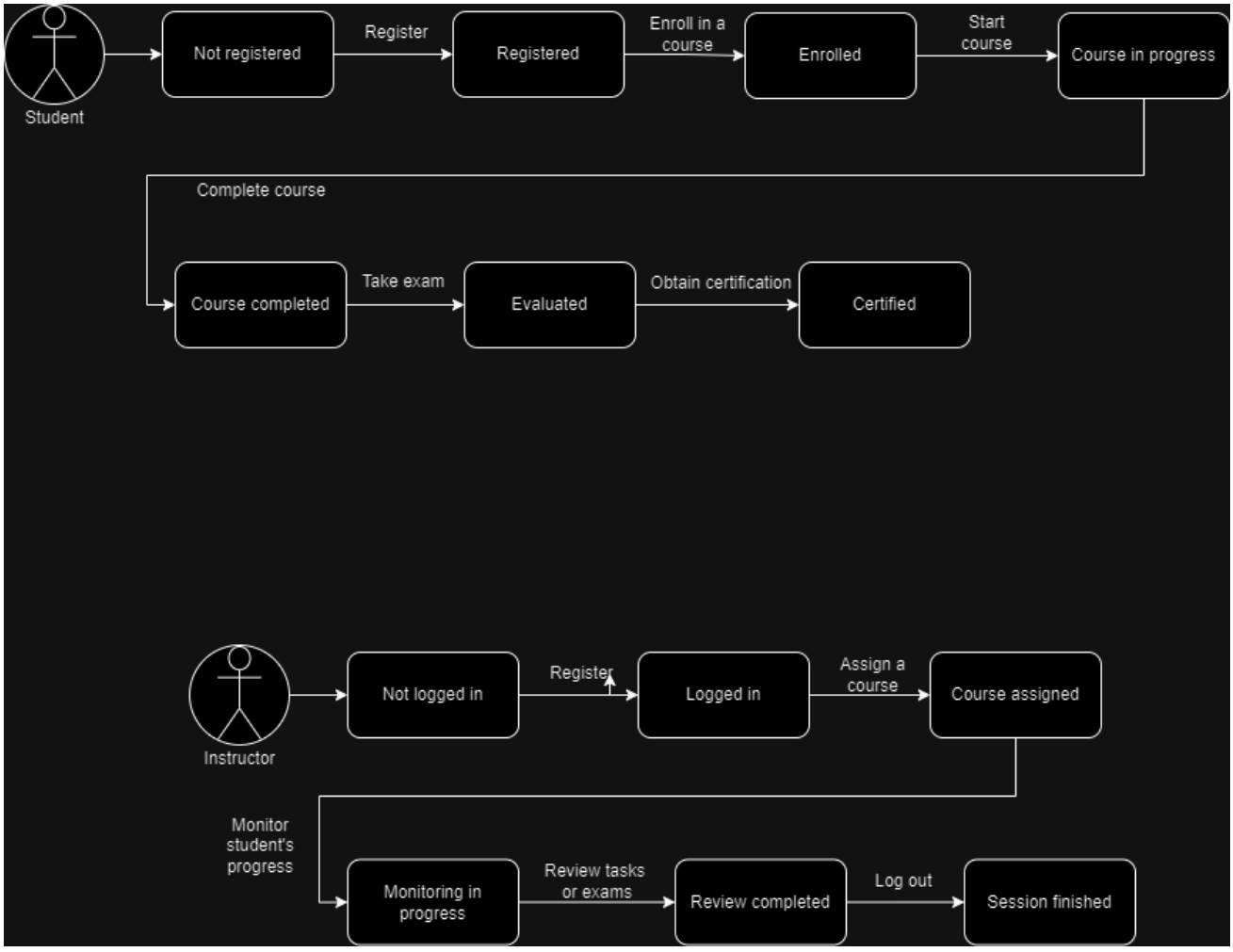
4. Appendices

Software Architecture Used:

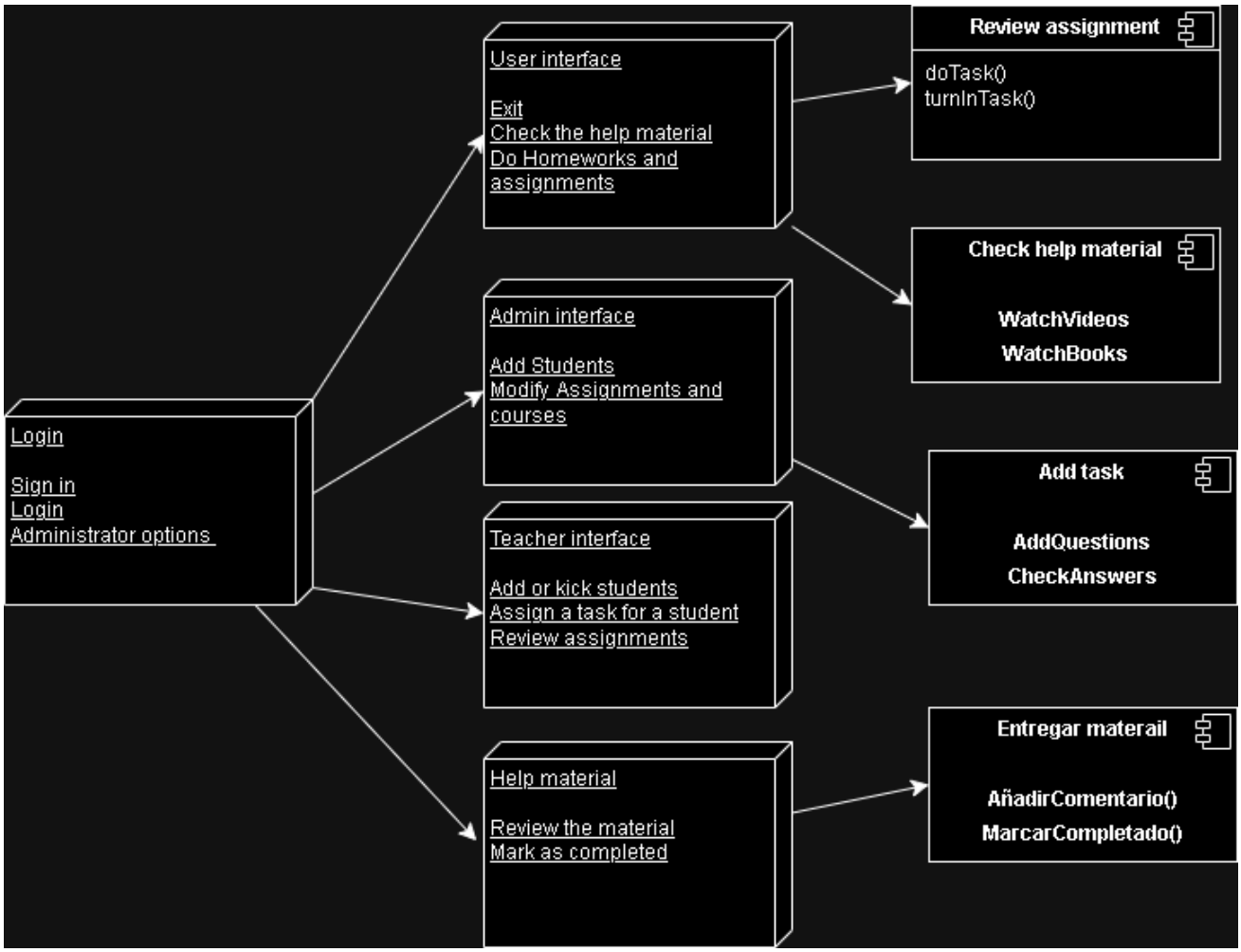
The architecture used is distributed, with the purpose of allowing votes to be viewed at different polling stations.

UML Diagrams

State diagrams

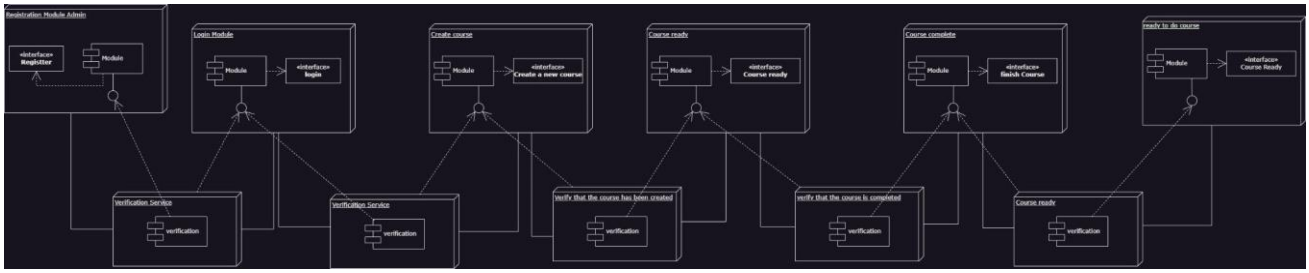


Deployment Diagram

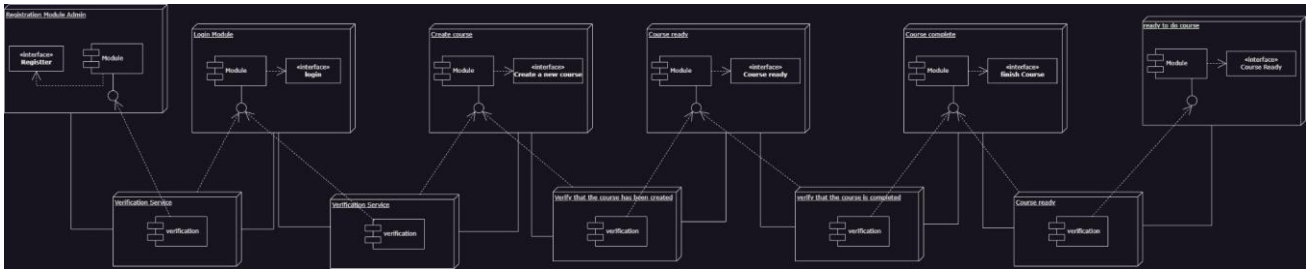

Login
Sign in
Login
Administrator options
User interface
Exit
Check the help material
Do Homeworks and assignments
Admin interface
Add Students
Modify Assignments and courses
Teacher interface
Add or kick students
Assign a task for a student
Review assignments
Help material
Review the material
Mark as completed
Review assignment
doTask()
turnInTask()
Check help material
WatchVideos
WatchBooks
Add task
AddQuestions
CheckAnswers
Entregar materail
AñadirComentario()
MarcarCompletado()

Component diagrams

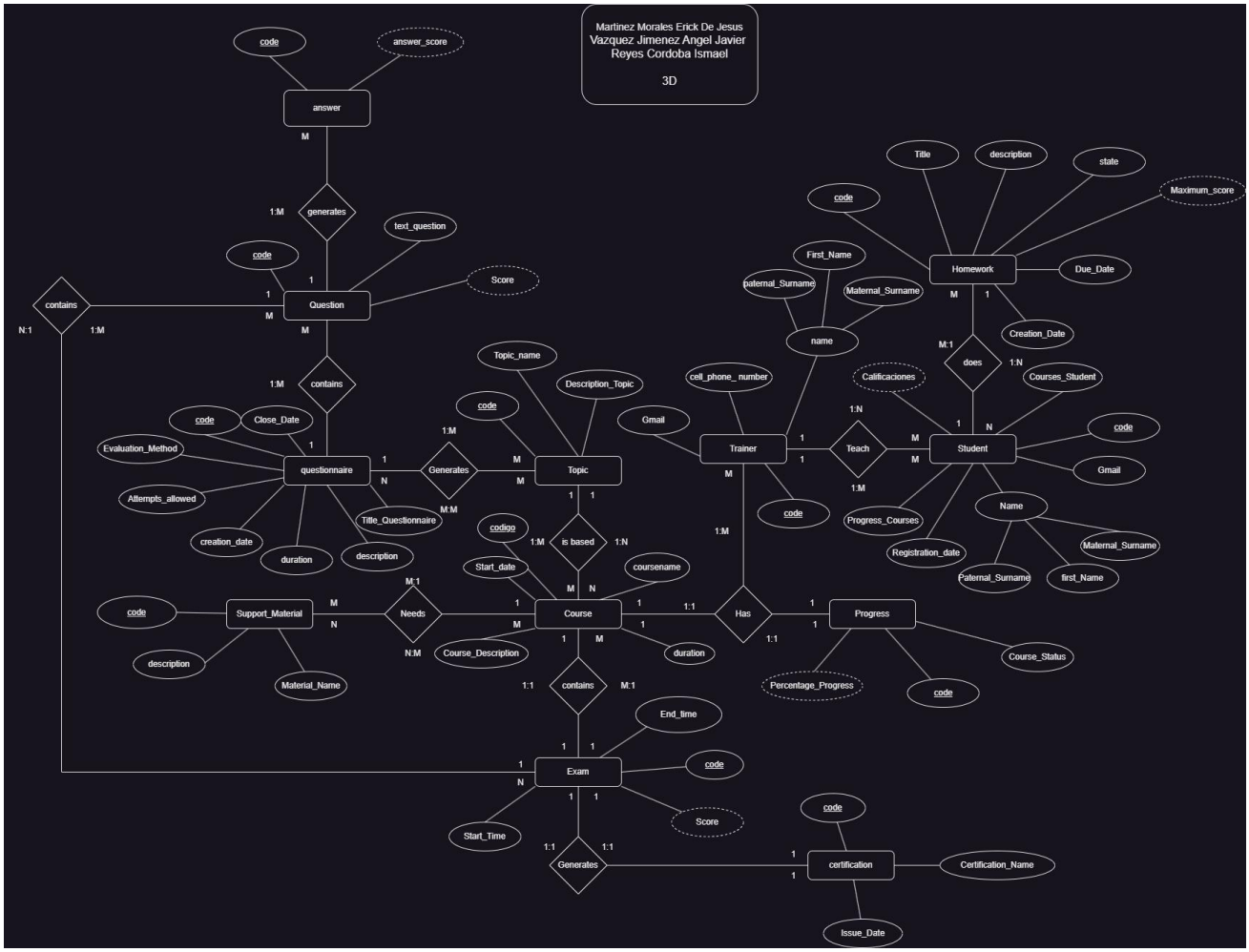
Manager



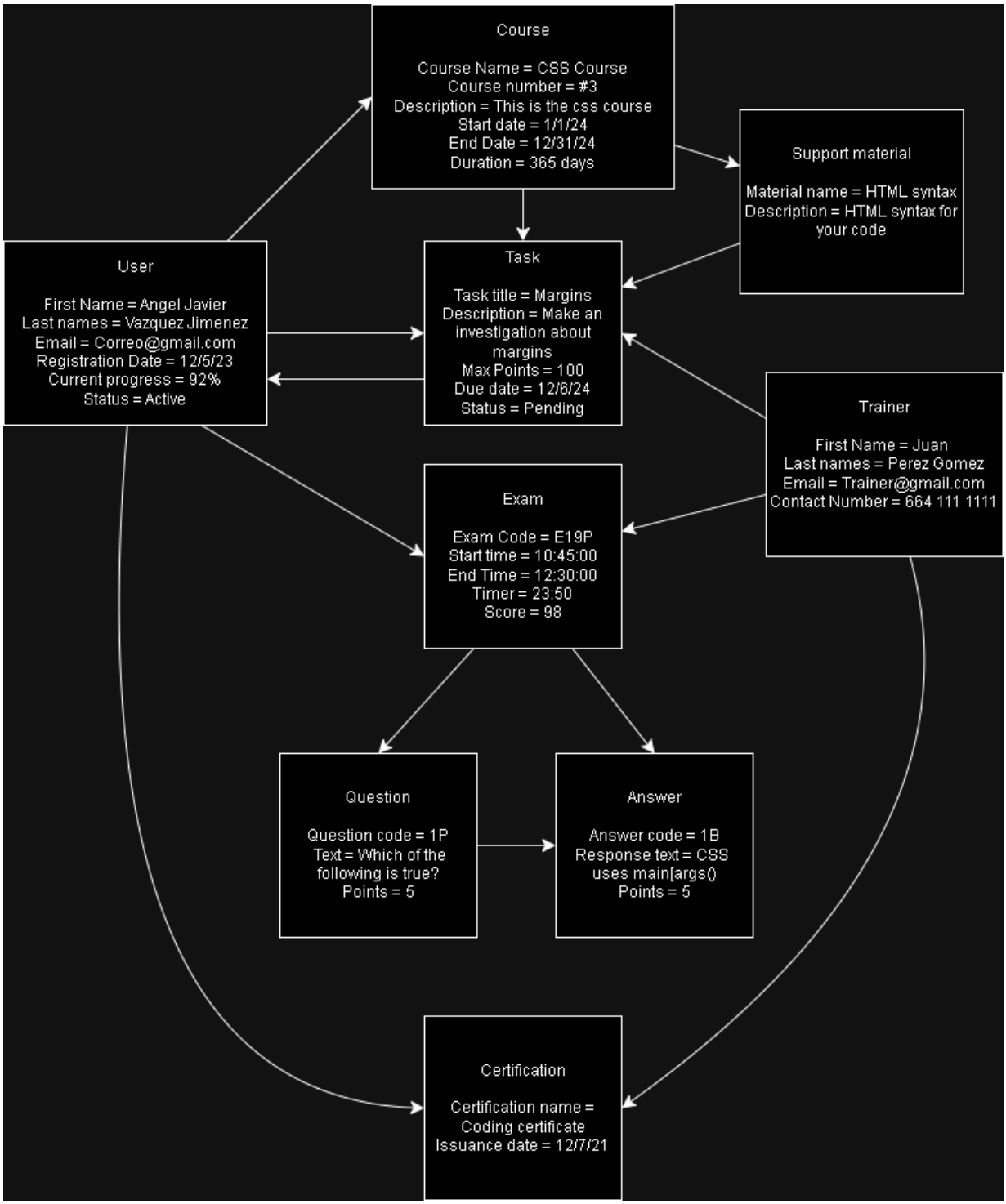
User



DER


Martinez Morales Erick De Jesus
Vazquez Jimenez Angel Javier
Reyes Cordoba Ismael
3D
code
answer_score
answer
M
1:M
generates
text_question
code
1
Question
Score
contains
N:1
1:M
M
1:M
contains
code
Close_Date
Evaluation_Method
questionnaire
Attempts_allowed
creation_date
duration
description
Title_Questionnaire
1
N
1:M
Generates
M
M
code
Topic_name
Description_Topic
Topic
1
1
codigo
1:M
is based
1:N
M
N
Start_date
coursename
Course
1
M
N:M
Course_Description
duration
1:1
contains
M:1
1
N
Exam
End_time
code
Score
Start_Time
1:1
1:1
Generates
code
Support_Material
description
Material_Name
M:1
M
N
Needs
M
N
1
1:1
Has
1
1
1:1
Progress
Course_Status
Percentage_Progress
code
1:M
code
Trainer
cell_phone_ number
Gmail
name
First_Name
paternal_Surname
Maternal_Surname
M
1
Teach
1:N
1:M
Califiaciones
Homework
Title
description
state
Maximum_score
code
Due_Date
Creation_Date
M:1
does
1:N
Courses_Student
1
N
Student
code
Gmail
Name
Maternal_Surname
first_Name
Paternal_Surname
Progress_Courses
Registration_date
certification
code
Certification_Name
Issue_Date

OBJECTS DIAGRAM



Code explanation:

Explanation of the Code

Database Connection: The code begins by including a database connection file (conect.php). It establishes a connection to the database using the connectDB() function.

Handling Form Submission:

The code checks if the request method is POST, indicating that the form has been submitted.

It retrieves user input from the form: full name, last names, cell phone number, username, and email.

Email Existence Check:

A prepared statement is used to check if the email already exists in the Cursante table.

If the email is found, an alert is displayed, and the user is redirected back to the registration page.

User Registration:

If the email does not exist, the current date is captured as the registration date.

A new prepared statement is created to insert the new user's details into the Cursante table, with fields for name, last names, cell number, email, registration date, and initialized course and grade values (both set to 0).

If the insertion is successful, an alert indicates the registration was successful, and the user is redirected to another page (proyecto.php).

If the insertion fails, an error message is shown, and the user is redirected back to the registration page.

HTML Form:

The HTML structure includes a form for user registration. Each input field is required, and there are labels for better accessibility.

A button is provided for form submission, and a link is included for users who already have an account, allowing them to log in.

Closing Resources:

After processing, the prepared statement and database connection are closed to free up resources.

Important Notes

Ensure that your database and the Cursante table are set up correctly to match the expected schema.

Input sanitization and validation are essential to avoid SQL injection attacks and to ensure the integrity of the data.

Consider adding password functionality for user accounts for better security in real applications.

Explanation of the Code

Database Connection: The code begins by including a database connection file (conect.php) to connect to the database using the connectDB() function.

Session Management: A session is started using session_start() to keep track of the user's logged-in state and store user data.

Handling Login Form Submission:

The code checks if the request method is POST, indicating that the login form has been submitted.

It retrieves the matricula (student ID) from the submitted form.

Checking Matricula:

A prepared statement is used to query the database for a record in the Cursante table that matches the provided matricula.

The bind_param method binds the matricula variable as an integer (assuming it is an INT).

Successful Login:

If a record is found, the corresponding student information is fetched using fetch_assoc().

The student's information is stored in the session as $_SESSION['cursante'].

A welcome message is displayed using an alert, and the user is redirected to the home page (inicio.php).

Matricula Not Found:

If no matching matricula is found in the database, an alert indicates that the matricula was not found, and the user is redirected back to the login page (proyecto.php).

HTML Form:

The HTML structure includes a simple login form with a single input field for the matricula.

A button for submitting the form is provided, and there is a link for users who do not have an account, directing them to the registration page.

Closing Resources:

After processing the login attempt, the prepared statement and database connection are closed to release resources.

Important Notes

Ensure the database and Cursante table are set up correctly, particularly that the Matricula field exists and is indexed.

Input validation and sanitization are crucial to prevent SQL injection attacks and ensure data integrity.

For enhanced security, consider implementing password authentication and secure session management practices.

Explanation of the Code

Function Definition:

The connectDB function is defined to establish a connection to a MySQL database using the mysqli extension. The return type of the function is specified as mysqli.

Database Connection:

The mysqli_connect() function attempts to connect to the database with the specified parameters:

"localhost": The hostname where the database server is located.

"root": The username used to connect to the database. In this case, it is set to the default MySQL user.

"": The password for the MySQL user, which is empty in this example.

"CursosDB": The name of the database to connect to.

Connection Check:

After attempting to connect, an if statement checks whether the connection was successful:

If successful, it outputs "Connected".

If not successful, it outputs "NOT Connected".

Return Value:

Regardless of the connection success or failure, the function returns the $db connection object. This allows other parts of the application to interact with the database.

Important Notes

Error Handling: While the function prints a message indicating whether the connection was successful, it’s generally better to handle errors more gracefully, especially in production environments. Instead of echoing messages, you could log the error and handle it accordingly.

Security: If this code is used in a public-facing application, consider using environment variables or configuration files to manage database credentials securely rather than hardcoding them in the script.

mysqli: This code uses the mysqli extension, which is a more secure and feature-rich alternative to the older mysql extension. Ensure that your PHP environment supports mysqli.

Explanation of the PHP Code

Database Connection Function (connectDB):

Establishes a connection to the MySQL database using mysqli_connect().

Returns the database connection object.

Form Submission Handling:

Checks if the request method is POST to handle form submissions.

Creating a New Section:

If sectionName is set in the POST request, it prepares an SQL statement to insert a new section into the Tema table.

Binds the section name and executes the statement.

Creating a New Task:

If taskName is set, it retrieves the task details and inserts them into the Tarea table with some default values for creation and due dates.

Creating a New Quiz:

If both quizTitle and questions are set, it prepares an SQL statement to insert a new quiz into the Examen table.

Then it iterates through the provided questions to insert them into the Pregunta table and links them to the quiz using the Examen_Pregunta table.

It also handles inserting answers for each question into the Respuesta table and linking them using Pregunta_Respuesta.

HTML Structure:

Basic HTML structure with sections for creating tasks, quizzes, and displaying existing sections and tasks.

Forms are provided for input, which will trigger JavaScript functions to manage the UI.

JavaScript Functions:

Functions for creating sections and tasks, updating the UI with the created sections and tasks, and managing quiz questions.

Alerts are used to notify users of successful operations.

This structured format helps maintain a clean separation between PHP backend logic and the frontend HTML/JavaScript interaction. Let me know if you need further assistance or modifications!

Explanation of the Code

HTML Structure:

The code uses standard HTML tags to define the page structure, including headers (<h2>, <h3>) and a list (<ul>) to display tasks.

Data Simulation:

An array of objects called tasks simulates tasks that might be retrieved from a server. Each task has an id, title, and status.

Function loadTasks():

This function populates the list of tasks in the HTML.

It clears the list before adding new tasks.

For each task in the array, it creates a new list element (<li>), which includes:

A header (<h4>) with the task title.

A button that allows the user to start the task if it is available.

Function startTask(taskId):

It displays an alert indicating that the user has started the corresponding task.

This function can be expanded to include additional logic, such as marking the task as "in progress".

Calling loadTasks():

This function is called when the page loads, ensuring that the list of tasks is displayed immediately.

This code allows users to view their tasks in the mathematics course and start those that are available. If you need further assistance or specific changes, feel free to let me know!